Pattern Recognition 131 (2022) 108849

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

Symbolic sequence representation with Markovian state optimization

Lifei Chen[a,b,*], Haiyan Wu[a], Wenxuan Kang[a], Shengrui Wang[c,*]

[a] College of Computer and Cyber Security & Digital Fujian Internet-of-Things Laboratory of Environmental Monitoring, Fujian Normal University, Fujian 350117, China
[b] Center for Applied Mathematics of Fujian Province, Fujian Normal University, Fujian 350117, China
[c] Department of Computer Science, Université de Sherbrooke, Quebec J1K 2R1, Canada

ARTICLE INFO

Article history:
Received 15 June 2021
Revised 15 April 2022
Accepted 12 June 2022
Available online 14 June 2022

Keywords:
Sequence representation
Hidden Markov model
State clustering
Hierarchical model selection
Activity recognition

ABSTRACT

Sequence representation, which is aimed at embedding sequentially symbolic data in a real space, is a foundational task in sequence pattern recognition. It is a difficult problem due to the challenges entailed in learning the intrinsic structural features within sequences in small sample size cases, in an unsupervised way. In this paper, we propose to represent each symbolic sequence by its transition probability distribution over discriminating topics, formalized by a set of optimized Hidden Markov Model (HMM) states shared by all sequences. An efficient method, called Markovian state clustering with hierarchical model selection, is proposed to optimize the Markovian states and to adaptively determine the number of topics. The proposed method is experimentally evaluated on human activity recognition and protein recognition, and results obtained demonstrate its effectiveness and efficiency.

1. Introduction

A symbolic sequence is a linear chain made up of events or symbols. Nowadays, such sequences are common in real-world applications, such as biological sequences in genomic studies and sequences of activities of daily living [1,2]. Compared to the standard scenario where each observation is a vector of features, learning from sequences is a more challenging problem, due to the difficulty of modeling the chronological dependencies of structural features and the possibly large variation in the length of the sequences [3,4].

Symbolic sequence representation, aimed at representing each sequence by a fixed-length vector that presents its structural features, is a foundational task in sequence pattern recognition [4,5]. Technically speaking, representing sequences by vectors is equivalent to embedding the sequential data in a vector space [6]. We are interested in the unsupervised method in this paper, because of its flexibility and the advantage that it requires no labeled data. Using such a method, therefore, the standard techniques originally designed for vector data can be easily transposed to symbolic sequences.

Existing methods can be roughly divided into two groups [2,3]: subsequential pattern-based embedding and model-based embedding. In the first group, the structural feature is explained as the occurrence of important patterns hidden in sequences. Such patterns, including the popular k-gram and frequent subsequence [7–9], are viewed as shallow features representing the sequences. However, the resulting representation model may be too high-dimensional or coarse-grained limiting their use in pattern recognition tasks [5]. Recently, there has been an explosion in the use of neural network-based auto-encoders such as the encoder-decoder LSTM (long short-term memory) [6] and CNN (convolutional neural network) [10]. Usually, a large number of samples are needed to train the neural networks.

Sequences are embedded into probability spaces with the methods in the second group, which have also been a subject of wide interest due to their clear probabilistic semantics and flexibility in small sample size cases. Usually, with these methods, a probability model is trained for each sequence, so that the dissimilarity between sequences can be measured based on differences between the model parameters [3,4]. In effect, this provides an *implicit* representation model for symbolic sequences. Popular among existing methods is the use of the hidden Markov model (HMM), which is a generative model based on the hypothesis that each sequence is conditioned on the state of a hidden Markov chain. They are thus more interpretable, as the hidden states of an HMM can be interpreted as topics within sequences [11,12].

While the HMM modeling method has been stressed extensively in sequence pattern recognition, previous studies focused mainly on parameter inference. Generally, such a method is designed to maximize the likelihood of sequences being observed

* Corresponding authors.
E-mail addresses: clfei@fjnu.edu.cn (L. Chen), shengrui.wang@usherbrooke.ca (S. Wang).

https://doi.org/10.1016/j.patcog.2022.108849
0031-3203/

from the model, given a predefined number of states [12–14]. We remark that the performance of HMM-based sequence representation depends heavily on the "quality" of the states themselves, which, however, are not guaranteed to be optimized in existing methods. Two key issues must be addressed here: the determination of the number of states, and the optimization of the states to produce a sufficiently discriminating representation [12,15]. These issues, which are directly related to several unsolved problems such as how to define the HMM distinguishability measures, remain significant challenges in representation studies.

In this paper, we formalize the above issues as a *Markovian state optimization* problem, and propose an *explicit* representation model for sequences based on the optimized HMM states. In a bit more detail, we propose to represent each symbolic sequence by its transition probability distribution over the underlying topics, which are defined as a set of discriminating HMM states shared by all sequences. To learn the optimized topics, a cluster-based adaptive method, called state clustering with hierarchical model selection, is proposed to discover the sequence topics from a model tree. We carry a series of experiments on commonly used sequence sets from real-world domains to demonstrate the performance of the proposed approach in comparison with popular methods including the neural network-based auto-encoder.

2. Related work and preliminaries

2.1. Pattern-based embedding

The pattern-based embedding methods have a long history in biological sequence analysis [2]. The popular method is to divide the long sequence into shorter fragments by sliding a window of length k (alternatively known as memory-length [8]), called a k-mer [5], k-gram [7] or k-tuple [9] in the literature. The patterns of interest (the resulting fragments) are considered as features spanning the new space. Since choosing an appropriate memory-length is currently an unsolved problem [8], a large memory-length is preferred in practice. However, the number of such patterns will increase exponentially with the length for a given sequence set, which means that the resulting vector space is generally in high-dimensionality.

The symbols appearing in the sequences can be viewed as words composing the sequences, analogous to the "documents" in natural language processing (NLP). According to this view, the numerous methods originally developed for NLP [12,14] can be applied to embed the sequences in a low-dimensional space. Before fed to deep learners such as the LSTM [6] and CNN auto-encoder [10], they have to be converted into numeric data, using the one-hot encoding or distributed representations such as Word2Vec [16]. Moreover, as the input sequences maybe in varying lengths, they have to be expanded to the maximal length or trimmed to a fixed length; this would make the learners unnecessarily complicated or lead to the information loss.

2.2. Model-based embedding

Markov models are frequently applied to symbolic sequence modeling, based on the Markov chain assumption that occurrence of each symbol in the sequence is solely related to its preceding subsequence of a fixed length. Generally, they are trained using the maximum likelihood estimation method, in which the sequential dependencies of the symbols within sequences are expressed by a set of conditional probability distributions (CPDs) [17].

In contrast to the traditional Markov models, hidden Markov models (HMMs) assume that the sequence observed is generated by a hidden state chain. A discrete-time HMM is characterized by two parameters L (the number of hidden states) and M (the number of distinct symbols per state), and three probability measure sets denoted by $\lambda = (\Pi, A, B)$. The initial state probability distribution Π represents the probability of the first state in the chain. Each row vector of the state transition matrix $A = [a_{ij}]_{L\times L}$ is the transition probability distribution of a state, where $a_{ij} = p(q_{t+1} = j|q_t = i)$ and q_t denotes the state at time t. In the state emission matrix $B = [b_{jm}]_{L\times M}$ with $b_{jm} = p(o_t = m|q_t = j)$, each row corresponds to the symbol probability distribution of a state; here, o_t denotes the observation at time t. Given a sequence s, the HMM is typically trained using an expectation-maximization (EM) algorithm such as the popular Baum-Welch-like algorithm to maximize the probability $p(s|\lambda)$ [4,13,14].

While it is known that hidden states of HMMs are meaningful in representing the underlying features of sequences [12,14], little effort has been made to train the HMM in order to optimize the states themselves. The major obstacles lie in difficulties in estimating the states to maximize the sequence likelihood while making the models distinguishable across different sequences [10], and in determining the optimal number of states [21,22]. The closest work related to the first issue is the discriminative training methods for HMMs. They usually aim at optimizing a new objective function, based on the discriminative criterion (e.g., the maximum mutual information) [18] or priori knowledge constraints [14,19]. For example, [14] suggests to train the shared-state HMMs with Dirichlet prior constraints, while [19] presents a gradient search method to identify the states by directly optimizing a set of domain-specific parameters closely related to the state aggregation. Note that the number of HMM states needs to be predefined in these methods.

Recently, the heuristic methods have been applied to estimate HMM parameters, including the number of states. Based on the ant colony [20] or particle swarm optimization [23], the best model can be obtained within a stochastic searching process, which, however, is generally time-consuming. Popular among the alternatives is the use of a wrapper, in which the number can be iteratively adjusted according to the model performance evaluated on the validation set. For example, the model is updated in each iteration using the linearly increased number [21] or a new one obtained by the Bakis's refinement [22]. Such a method is thus also inefficient and, more importantly, is difficult to adapt to the unsupervised scenario.

The problem of determining the number of states can be viewed as a model selection problem, where a set of candidate HMMs with different number of states is evaluated by a validation criterion and the number corresponding to the optimal value is returned [9]. Typically, the criterion is defined based on the penalty for the model complexity, such as the revised BIC [24] and the marginal log-likelihood in the variational Bayesian inference model [25]. Using such a criterion, however, the selected model would yield a biased indication of the discrimination between HMM states. In our method as presented below, the number of states can be determined efficiently in a hierarchical model selection procedure, using a new validation criterion directly defined on the discriminative quality of the states.

3. Probabilistic representation of symbolic sequences

In this section, our new sequence representation model is presented, based on which the state optimization problem and the new representation algorithm are defined.

3.1. K-topic generative model

In what follows, the sequence set to be learned is denoted by $\mathcal{S} = \{s_n|1 \le n \le N\}$, where $s_n = o_{n1} \dots o_{nt} \dots o_{nT_n}$ stands for the n-th sequence of length T_n and N for the number of sequences. Each

symbol in the sequence, o_{nt} for $t = 1, 2, \ldots, T_n$, is taken from an alphabet consisting of M symbols, and we will use $m = 1, 2, \ldots, M$ for indexing.

We suppose that each sequence $s_n \in \mathcal{S}$ is an observation of the hidden chain $q_{n1} \ldots q_{nt} \ldots q_{nT_n}$, and each node on the chain is one of the K discriminating topics from the set $\mathcal{C} = \{c_k | 1 \le k \le K\}$, i.e., $q_{nt} \in \mathcal{C}$. Here, the term *topics* is an alias for the underlying patterns or deep structural features shared by all sequences in $\mathcal{S}$. The emission of observations from the topics is governed by the matrix $\tilde{B}$, in which each row vector represents the topic distribution over the symbols. Therefore, it is $\tilde{B}$ and the hyper-parameter K that define the topics.

Through such modeling, the difference between sequences will be indicated by the difference in the distribution of topics in their respective topic chains. This is a simulation for many application scenarios. In classifying the $33(\alpha/\beta)_8$-barrel proteins [9], for example, the proteins must be subdivided according to their enzymatic activities (such as the "β-galactosidase"). Each activity here can be modeled as one "topic" associated with the biological function expressed by the various catalytic modules in the sequences. The proteins are different from each other due to the individual periodic character of their catalytic modules (here, different occurrence of the topics).

We further suppose that the topics chain is one kind of first-order time-homogeneous Markov chain. That is, for s_n, the topic occurring in q_{nt} is conditioned solely on its occurrence in $q_{n(t-1)}$ at any time $t > 1$, and the probability of the transition is independent of time. Thus, the topic transitions in the chain can be modeled by the CPDs [17], given by $\tilde{A}_n = [\tilde{a}_{kj}^{(n)}]_{K\times K}$, where $\tilde{a}_{kj}^{(n)} = p(c_j|c_k; s_n)$ is the probability that the topic changes from c_k to c_j at two consecutive times, satisfying

$$\forall 1 \le k, j \le K, n : \tilde{a}_{kj}^{(n)} \ge 0, \text{ and } ||\tilde{\mathbf{a}}_k^{(n)}||_1 = 1. \tag{1}$$

Here, the row vector $\tilde{\mathbf{a}}_k^{(n)} = (\tilde{a}_{k1}^{(n)}, \tilde{a}_{k2}^{(n)}, \ldots, \tilde{a}_{kK}^{(n)})$ gives the conditional probability distribution of the k-th topic within s_n. Clearly, it is $\tilde{A}_n$ that distinguishes the sequence s_n from the others, given the K topics.

3.2. Probabilistic representation

The CPD matrix $\tilde{A}_n$ is somewhat similar to the transition matrix in the standard HMM, if we map the topics with the HMM states. The major difference is that our model requires the topics to be discriminative and the number of topics K to be optimized. Also, our CPD matrices differ between sequences, but the emission probabilities for each topic ($\tilde{B}$) remain the same.

We thus propose to represent each sequence s_n by its CPD matrix $\tilde{A}_n$ in the K-topic model discussed in the last subsection. With such a representation, the similarity between two sequences s_n and $s_{n'}$ can be defined as the sum of the pairwise similarities of the K CPDs in $\tilde{A}_n$ and $\tilde{A}_{n'}$, respectively. The latter can be measured using a probability product kernel [26], such as the *Bhattacharyya kernel* given as $\kappa(u, v) = \sum_x \sqrt{u(x)v(x)}$, where u and v are discrete probability distributions over the same domain. In our case, the similarity between s_n and $s_{n'}$ is thus computed as $\sum_{k=1}^{K} \kappa(\tilde{\mathbf{a}}_k^{(n)}, \tilde{\mathbf{a}}_k^{(n')}) = \sum_{k=1}^{K}\sum_{j=1}^{K} \sqrt{\tilde{a}_{kj}^{(n)}\tilde{a}_{kj}^{(n')}}$. This suggests a probabilistic representation for symbolic sequences, as set out in Definition 1 below.

Definition 1. (K-topic representation) The K-topic representation of sequence s_n is the matrix denoted by

$$\mathbb{R}(s_n) = \left[\sqrt{\tilde{a}_{kj}^{(n)}}\right]_{1\le k,j\le K.}$$

Based on Definition 1, the dissimilarity between s_n and $s_{n'}$ can be computed using a divergence measure, such as $||\mathbb{R}(s_n) - \mathbb{R}(s_{n'})||_2^2$, which yields

$$\mathbb{D}(s_n, s_{n'}) = \sum_{k=1}^{K}\sum_{j=1}^{K}\left(\sqrt{\tilde{a}_{kj}^{(n)}} - \sqrt{\tilde{a}_{kj}^{(n')}}\right)^2 = 2\sum_{k=1}^{K}(1 - \kappa(\tilde{\mathbf{a}}_k^{(n)}, \tilde{\mathbf{a}}_k^{(n')})).$$

The resulting equation is precisely the squared *Hellinger distance* between discrete probability distributions [27]; we will return to this in Section 4.2.

3.3. HMM-based learning algorithm

The key issue in generating the K-topic model is the learning of K high-quality topics from $\mathcal{S}$. For the purpose, we define the topics as a set of optimized HMM states that maximize some criteria related to the topic quality. In this case, the topics will be mapped to a specific set of HMM states, which are shared by all sequences and associated with an optimized emission probability distribution, $\tilde{B}$, called *topic distribution* in the following. Definition 2 below formulates the optimization problem.

Definition 2. (State optimization) The Markovian state optimization problem entails learning the topic distribution $\tilde{B}$, such that the quality (denoted by Q and formally defined in Section 4.3) is maximized, i.e.,

$$(K, \tilde{B}) = \arg\max_{\forall(\hat{K},\hat{B})} Q(\hat{K}, \hat{B}).$$

Based on the state optimization, it is possible to derive our representation method, as illustrated in Fig. 1, which comprises three stages. First, we train an HMM for each sequence $s_n \in \mathcal{S}$ with the same L initial states. Therefore, $H = N \cdot L$ hidden states can be obtained in the trained L-state HMMs. We denote the transition and emission matrix of s_n by $A_n^{(L)}$ and $B_n^{(L)}$, respectively, and the collection of all the states by $\mathcal{V}$, i.e., $\mathcal{V} = \{B_n^{(L)} | 1 \le n \le N\}$. Next, we consider $\mathcal{V}$ as the candidate topics and solve the state optimization problem defined in Definition 2, to obtain the K topics. In the third stage, the HMMs are re-trained by fixing the state emission distribution for the K topics; thus, a new transition matrix (say, $A_n^{(K)}$) can be obtained from the re-trained model for each $s_n \in \mathcal{S}$. We finally use the transition matrix for the K-topic representation, i.e.,

$$\tilde{A}_n = A_n^{(K)}. \tag{2}$$

The learning algorithm, named sopHMM (short for state-optimized HMM), is described in Algorithm 1. Note that the number of topics (K) is not a predefined parameter, but a hyper-parameter jointly learned along with the topic distribution in $\tilde{B}$; both will be discussed in Section 4, where we can see that $K << H$. Therefore, the time complexity of the algorithm is dominated by the HMM training method used in Step 1 (as well as Step 3). When using the Baum-Welch reestimation algorithm [3], the time complexity for each sequence is $O(TL^2)$, given that generally $L \le M$ [12]. The time complexity of Algorithm 1 is thus $O(NTL^2)$.

Algorithm 1 Outline of the sopHMM algorithm for Sequence representation.

Require: sequence set $\mathcal{S}$;
Ensure: K-topic representation of $\mathcal{S}$, i.e., $\{\mathbb{R}(s_n)|s_n \in \mathcal{S}\}$;
1: Train an HMM for each $s_n \in \mathcal{S}$, and collect all the hidden states into $\mathcal{V}$.
2: Obtain the number of topics, K, and the topic set, $\tilde{B}$, by calling Algorithm 2, i.e., $(K, \tilde{B})$ = StateOptimization($\mathcal{V}$).
3: Re-train the HMMs with the K topics in $\tilde{B}$, and obtain each $\mathbb{R}(s_n)$ according to Eq. (2) and Definition 1.

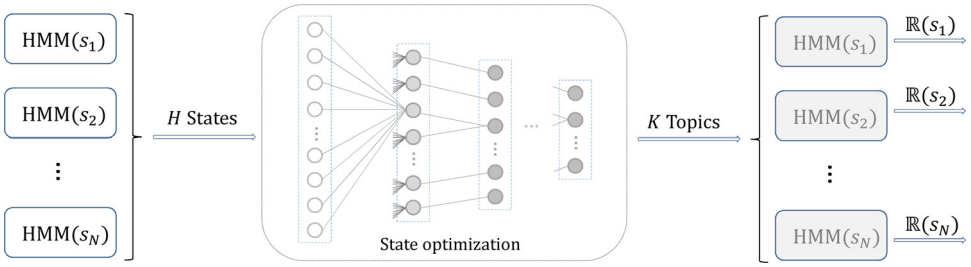


Fig. 1. Illustration of our HMM-based sequence representation learning method.

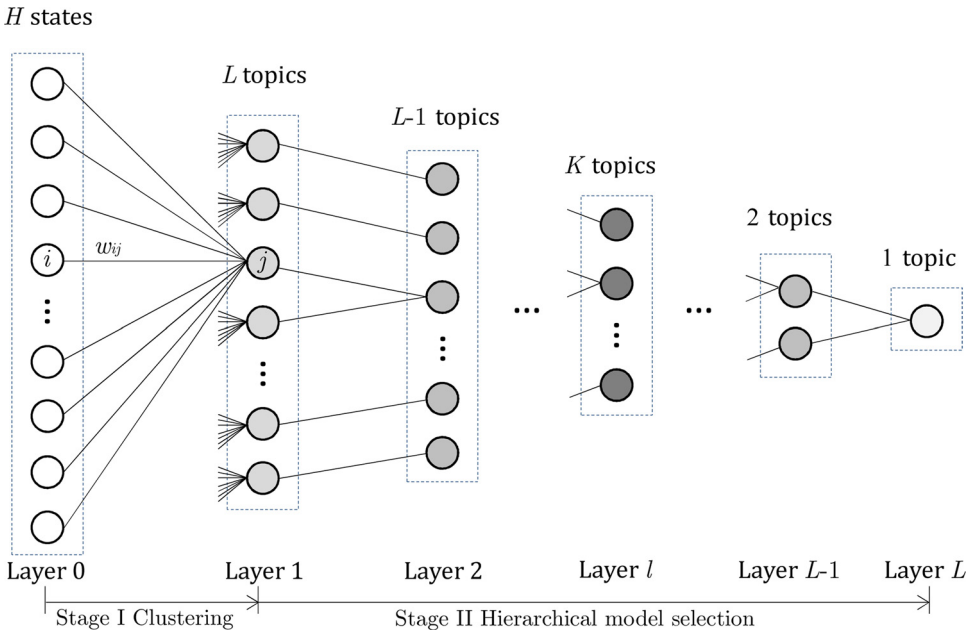


Fig. 2. Illustration of the two-stage state optimization method and the model tree.

4. Markovian state optimization

The aim of this section is to learn K topics from the H states with $K << H$, by solving the optimization problem defined in Definition 2. Recall that the topics are defined as a set of optimized states, and both the topics and the states are represented by emission probability distributions over the M symbols. The task can thus be understood as an *adaptive* clustering problem, in which the input H states are grouped into K homogeneous clusters (topics) while making the clusters distinguishable with the number of clusters K automatically determined. Next, we begin by outlining the new method.

4.1. Outline of the state optimization method

Generally, the adaptive clustering problem is solved by hierarchical clustering in the literature [28,29]. However, the hierarchical clustering algorithm typically have a high time-complexity, reaching $O(H^2)$ in our case where $H = N \cdot L$. To derive an efficient method, we propose a two-stage solution, as illustrated in Fig. 2.

The goal of Stage I in our method is to reduce the number of inputs, which are then fed to the second stage (Stage II) for *hierarchical model selection*. The candidate topics obtained in the two stages can be viewed as a model tree consisting of L layers. Each node in the input layer, Layer 0, corresponds to a state in $\mathcal{V}$, while the topics are distributed on different layers starting from Layer 1. The nodes in Layer 1 stand for the L initial candidate topics (generated by Stage I with $L << H$), each being a special combination of the states in Layer 0. On Layer l, where $1 \le l < L-1$, the two closest topics are merged to create a new topic for the next layer (Stage II). Therefore, there will be $L-1$ layers, each consisting of $L, L-1, \ldots, 2$ admissible topics.

We remark that the two-stage method solves the optimization problem in Definition 2 with a divide-and-conquer strategy. From the perspective of *cluster validity*, Stage I produces L high-quality candidate topics in terms of the within-cluster compactness, that is, topic homogeneity; while Stage II aims at choosing from the candidates a K-component model that maximizes the between-cluster separation, corresponding to the topic distinguishability. Given the candidates in Layer 1, the model selection problem thus becomes the new problem of selecting from the model tree a specific layer having the highest between-cluster separation, called *topic scatter* in the following.

Formally, we denote the model on Layer l by $(\hat{K}_l, \hat{B}_l)$, where $\hat{K}_l = L - l + 1$ is the number of topics in the layer and $\hat{B}_l = \{\hat{\mathbf{b}}_j | 1 \le j \le \hat{K}_l\}$ the set of emission probability distributions for the topics. Here, $1 \le l \le L-1$ and $\hat{\mathbf{b}}_j = (\hat{b}_{j1}, \hat{b}_{j2}, \ldots, \hat{b}_{jM})$ satisfying

$$\forall j : ||\hat{\mathbf{b}}_j||_1 = 1. \tag{3}$$

We then measure the quality of the topics on Layer l by $Q(\hat{K}_l, \hat{B}_l)$; consequently, the optimization problem in Definition 2 is rewritten as

$$(K, \tilde{B}) = \arg\max_{1 \le l \le L-1} Q(\hat{K}_l, \hat{B}_l). \tag{4}$$

The hierarchical model selection algorithm is outlined in Algorithm 2. First, the algorithm creates L initial topics from

Algorithm 2 Cluster-based HMM state optimization (StateOptimization).

Require: states set $\mathcal{V}$;
Ensure: number of topics K, topics set $\tilde{B}$;
1: Group $\mathcal{V}$ into L clusters, and obtain $\hat{B}_1$ from the cluster centroids;
2: Build Layer 1 of the model tree from the model $(L, \hat{B}_1)$;
3: **for** $l = 2$ to $L-1$ **do**
4: Update $\hat{B}_{l-1}$ to $\hat{B}_l$ with the two closest topics in $\hat{B}_{l-1}$ merged;
5: Build Layer l of the tree from the model $(L-l+1, \hat{B}_l)$.
6: **end for**
7: Output K and $\tilde{B}$ by model selection according to Eq.~(4).

the H input states (using a clustering method presented in the next subsection), and build Layer 1 of the model tree. Then, the remaining layers are built one after another by means of hierarchical agglomeration; see Section 4.3 for details. Finally, the optimal model is obtained according to the rule of Eq. (4).

4.2. State clustering

In this subsection, we move to define an algorithm for Stage I to generate the initial topics. The input is given by $\mathbf{v}_i = (v_{i1}, v_{i2}, \ldots, v_{iM}) \in \mathcal{V}$ for $i \in [1, H]$, and the output is denoted by $\hat{B}_1 = \{\hat{\mathbf{b}}_j | 1 \le j \le L < H\}$ subject to Eq. (3). Note that both $\mathbf{v}_i$ and $\hat{\mathbf{b}}_j$ are represented by an M-length vector, which is interpreted as a probability distribution over the M symbols. We model the resulting L topics as the L cluster *centroids* yielded by a partition-based clustering algorithm. Besides, the relevance of each state to the topic, shown as w_{ij} in Fig. 2, is exactly the cluster membership therein. We formulate these in a probabilistic clustering model, as follows:

It is assumed that each input $\mathbf{v}_i \in V$ is independently and identically distributed from the mixture density population:

$$F(\mathbf{v}_i; \Theta) = \sum_{j=1}^{L} \alpha_j \times G(\mathbf{v}_i | \hat{\mathbf{b}}_j, \sigma),$$

where $\Theta = \{\sigma, \alpha_j, \hat{\mathbf{b}}_j | 1 \le j \le L\}$ and α_j is the mixing coefficient. Here, $G(\cdot|\hat{\mathbf{b}}_j, \sigma)$ is a Gaussian density function describing the probability distribution of topic $\hat{\mathbf{b}}_j$ (i.e., the cluster centroid), given by

$$G(\mathbf{v}_i | \hat{\mathbf{b}}_j, \sigma) = (2\pi\sigma^2)^{-\frac{M}{2}} \exp\left(-\frac{1}{\sigma^2}(1 - \kappa(\mathbf{v}_i, \hat{\mathbf{b}}_j))\right) \tag{5}$$

where σ is the variance of the Gaussian. Note that Eq. (5) is an extension to the standard Gaussian function, because here both $\mathbf{v}_i$ and $\hat{\mathbf{b}}_j$ are essentially a probability distribution, not a multi-dimensional vector originating from the standard multivariate Gaussian mixture. The difference lies in the measure used for the sample-to-mean variation, which is $2(1 - \kappa(\mathbf{v}_i, \hat{\mathbf{b}}_j))$ in Eq. (5) measuring the distance between two probability vectors. The measure used here is precisely the squared Hellinger distance as discussed in Section 3.2. Moreover, we formulate the relevance of the input state $\mathbf{v}_i$ to the topic $\hat{\mathbf{b}}_j$, denoted by w_{ij}, as the posterior probability,

$$w_{ij} = p(\hat{\mathbf{b}}_j | \mathbf{v}_i) \triangleq \frac{\alpha_j \times G(\mathbf{v}_i | \hat{\mathbf{b}}_j, \sigma)}{F(\mathbf{v}_i; \Theta)} \tag{6}$$

$$\text{s.t.} \quad \forall i, j : \sum_{j=1}^{L} \alpha_j = 1, \text{ and } \sum_{j=1}^{L} w_{ij} = 1. \tag{7}$$

By applying the mixture model to the state clustering problem, the goal is to estimate Θ from the inputs to minimize the expected logarithmic likelihood over all the topics, i.e.,

$$\begin{aligned} \min_{\Theta} \quad & -\sum_{i=1}^{H} \sum_{j=1}^{L} p(\hat{\mathbf{b}}_j | \mathbf{v}_i) \ln F(\mathbf{v}_i; \Theta) \\ = & \sum_{i=1}^{H} \sum_{j=1}^{L} w_{ij} \ln \frac{w_{ij}}{\alpha_j \times G(\mathbf{v}_i | \hat{\mathbf{b}}_j, \sigma)} \qquad \text{(by Eq. (6))}. \end{aligned}$$

Substituting for G according to Eq. (5), the clustering objective function that needs to be minimized can be obtained as (ignoring the constants):

$$J(\Theta) = \sum_{i=1}^{H} \sum_{j=1}^{L} w_{ij} \times \left(\frac{1 - \kappa(\mathbf{v}_i, \hat{\mathbf{b}}_j)}{\sigma^2} + \ln \frac{w_{ij}\sigma^M}{\alpha_j}\right) \tag{8}$$

subject to Eqs. (3) and (7). We have the following result.

Theorem 1. *J is convex with respect to $\sigma, \alpha_j, \hat{b}_{jm}$ and w_{ij} for $i \in [1, H]$, $j \in [1, L]$, $m \in [1, M]$, respectively, and is minimized when*

$$w_{ij} = \frac{\alpha_j \exp(-\frac{1}{\sigma^2}(1 - \kappa(\mathbf{v}_i, \hat{\mathbf{b}}_j)))}{\sum_{j'=1}^{L} \alpha_{j'} \exp(-\frac{1}{\sigma^2}(1 - \kappa(\mathbf{v}_i, \hat{\mathbf{b}}_{j'})))}, \tag{9}$$

$$\hat{b}_{jm} = \frac{(\sum_{i=1}^{H} w_{ij}\sqrt{v_{im}})^2}{\sum_{m'=1}^{M}(\sum_{i=1}^{H} w_{ij}\sqrt{v_{im'}})^2}, \tag{10}$$

$$\alpha_j = \frac{1}{H}\sum_{i=1}^{H} w_{ij}, \tag{11}$$

$$\sigma = \frac{2}{HM}\sum_{j=1}^{L}\sum_{i=1}^{H} w_{ij} \times (1 - \kappa(\mathbf{v}_i, \hat{\mathbf{b}}_j)). \tag{12}$$

Proof. The proof is given in Appendix A.

Based on Theorem 1, a k-means-type algorithm can be derived to optimize the objective function, by learning the parameters iteratively in a sequential structure. The greedy method suggested by [9] is applied to initialize the initial topics in $\hat{B}_1$. Then, in each iteration, w_{ij}s, $\hat{b}_{jm}$s, α_js and σ are updated in turn according to Eqs. (9), (10), (11) and (12), respectively. Finally, $\hat{B}_1$ is output when convergence is reached. In practice, we asserted the convergence by examining whether the change in the posterior probabilities between two successive iterations is smaller than 10^{-4}. The time complexity of the state clustering algorithm is thus $O(LH)$ or equivalently $O(NL^2)$. □

4.3. Hierarchical model selection

To measure the quality of the model $(\hat{K}_l, \hat{B}_l)$ on Layer l in terms of the topic scatter, we suggest the following definition by averaging the deviation between each topic distribution in $\hat{B}_l$ and the uniform distribution. It can be seen that the greater the average topic-scatter, the more scattered the topics in a model. As our aim is to learn a set of discriminating topics, a model with larger Q should be preferred.

Definition 3. (Average topic-scatter) Letting $\mu = (\frac{1}{M}, \frac{1}{M}, \ldots, \frac{1}{M})$ be the uniform distribution over the M symbols, the average topic-scatter of the model $(\hat{K}_l, \hat{B}_l)$ is measured as

$$\begin{aligned} Q(\hat{K}_l, \hat{B}_l) \quad & = \frac{1}{\hat{K}_l}\sum_{j=1}^{\hat{K}_l}(\hat{\mathbf{b}}_j - \mu)(\hat{\mathbf{b}}_j - \mu)^\top \\ & = \frac{1}{\hat{K}_l}\sum_{j=1}^{\hat{K}_l}||\hat{\mathbf{b}}_j||_2^2 - \frac{1}{M} \sim \frac{1}{\hat{K}_l}\sum_{j=1}^{\hat{K}_l}||\hat{\mathbf{b}}_j||_2^2. \end{aligned}$$

Using the L initial topics yielded by the state clustering algorithm, the model tree can be built by hierarchically merging the two most similar topics on each layer. For Layer l where $1 \le l < L-1$ and thus $\hat{K}_l > 2$, we choose the two topics $\hat{\mathbf{b}}_1$ and $\hat{\mathbf{b}}_2$, which need to be merged according to

$$(\hat{\mathbf{b}}_1, \hat{\mathbf{b}}_2) = \arg\max_{1 \le j \ne j' \le \hat{K}_l} \kappa(\hat{\mathbf{b}}_j, \hat{\mathbf{b}}_{j'}).$$

Here, the Bhattacharyya kernel is used again to measure the similarity between topics. They are merged to create a new topic, $\mathbf{e} = (e_1, \ldots, e_m, \ldots, e_M)$, using a method similar to that used in the clustering algorithm. We view $\hat{\mathbf{b}}_1$ and $\hat{\mathbf{b}}_2$ as the two states being assigned to the cluster $\mathbf{e}$. After that, by fixing the posterior

Table 1
Details of the sequence sets used in the experiments.

Domain	Dataset	# Classes	M	N	T	Average T
Human activity recognition	ADL	2	9	35	[17,47]	27
	JSI	2	9	100	[2,50]	18
	LIBRAS	15	9	360	[44,44]	44
Protein recognition	33 $(\alpha/\beta)_8$-barrel	5	21	33	[598,1270]	872
	COG-S1	10	20	1000	[109,1162]	343
	COG-S2	20	20	2000	[53,1254]	369

probabilities at 1, the expression for updating the cluster centroid, Eq. (10), can be reused here, yielding

$$e_m = \frac{(\sum_{i=1}^{2} 1 \times \sqrt{\hat{b}_{im}})^2}{\sum_{m'=1}^{M}(\sum_{i=1}^{2} 1 \times \sqrt{\hat{b}_{im'}})^2} = \frac{(\sqrt{\hat{b}_{1m}} + \sqrt{\hat{b}_{2m}})^2}{2(1+\kappa(\hat{\mathbf{b}}_1, \hat{\mathbf{b}}_2))}.$$

Now, it is possible to generate the model for Layer $l+1$, which is $(\hat{K}_l - 1, \hat{B}_{l+1})$ with $\hat{B}_{l+1} = \{\mathbf{e}\} \bigcup \hat{B}_l / \{\hat{\mathbf{b}}_1, \hat{\mathbf{b}}_2\}$.

Building the model tree based on the above method has a time complexity of $O(L^2)$, as the tree contains L leaves in Layer 1. Recall that the state clustering algorithm used to generate the layer is $O(NL^2)$ in time (see Section 4.2). Therefore, the time complexity of Algorithm 2 is $O(NL^2)$ on the whole. Since it serves as Step 2 of our representation learning algorithm sopHMM (Algorithm 1), in which the other steps are $O(TNL^2)$ in time, we conclude that sopHMM has a time complexity of $O(TNL^2)$.

5. Experimental evaluation

In this section, the performance of our method (sopHMM) is evaluated on real-world sequence sets, and we also experimentally compare sopHMM with a few representative methods for sequence representation.

5.1. Datasets

Sequence sets from two real-world domains: human activity recognition and protein recognition in computational biology, were used in our experiments. Table 1 lists the details. The three datasets used for human activity recognition were obtained from the UCI machine learning repository. The first is the ADL (Activities of Daily Living) set [30], which comprises ADLs performed by two subjects in 35 days. Each sequence in the set is a series of daily living events picked from some binary sensors. As their homes had different sensors, we used solely the nine shared events shown in Table 3.

We also used the localization data for posture reconstruction provided by Jozef Stefan Institute (JSI) [31] for fall detection. The database was created from 5 persons wearing four sensors, each recording an indefinite number of moving points. The dataset used in the experiments contains 100 instances corresponding to two easily-confused activities: *falling* and *lying down*. The x-z coordinates of the chest sensor were converted into symbolic sequences as follows: First, the coordinates were rescaled to the range [0,1] and the resulting data space was divided into 50×50 grids. Then, starting from the second point, the movement direction of each point relative to the previous one was traced. The sequence was finally constructed by collecting the movements, each being a symbol representing one of the 9 directions.

The third set was obtained from the LIBRAS Movement Database [32], which was created for hand movement recognition. The database contains 15 classes of 24 instances each, where each class references a hand movement type such as Circle, Vertical zigzag, and Face-up curve, etc. The hand movement is originally represented by 90 features, which are coordinate series of 45 consecutive points on a bi-dimensional space. In our experiments, each coordinate series was converted into a symbolic sequence of length 44, using the method similar to that used for JSI.

Compared with the data used for human activity recognition, the protein sequences contain more symbols (20 standard amino acids). The first sequence set for protein recognition contains 33 $(\alpha/\beta)_8$-barrel proteins belonging to Glycoside Hydrolase family 2 from the CAZy database, which was discussed as an example in Section 3.1 and dummyTXdummy- was also studied in [9]. Although this set has only 33 proteins, the average sequence-length reaches 872 and 5 classes need to be distinguished (namely, Ga for β-galactosidase, Gi for β-glucuronidase, Cs for exo-β-D-glucosaminidase, Ma and Un for β-mannosidase). Sequences in this set contain an additional symbol.

Proteins encoded in complete genomes, commonly known as the Clusters of Orthologous Groups (COG) of proteins database[1], were also used for tests. We randomly generated 2 different subsets, named COG-S1 and COG-S2, from the COG database. As Table 1 shows, these two protein sets have a larger number of classes and sequences, and the length of the sequences changes significantly.

5.2. Experimental setup

The performances of different representation methods were evaluated by their use for sequence classification. Two classical classifiers, the support vector machine (SVM) and the prototype-based classifier (PBC), were chosen for the experiments. A brief presentation of these is given below:

SVM: we used a linear SVM [33] for the evaluation. In principle, representing sequences by vectors equates to projecting them onto a new space where the sequences belonging to different classes are expected to be well separated; in this case, a linear SVM would achieve high accuracy.

PBC, alternatively known as centroid-based classification [34], classifies a sample into its nearest prototype, i.e., the class center (centroid). Such a classifier can thus be used for evaluation in terms of maintaining the underlying patterns in the new representations.

Both classifiers work on the vector data sets generated by the sequence representation methods. In the experiments, each data set is classified by each classifier 20 times, each with a five-fold cross-validation. In this way, the performance can be quantified by the classification quality, for which we have adopted two approaches. One is the common *classification accuracy* (CA) which calculates the proportion of samples correctly classified. The other approach is based on the use of the F1 measure; in particular, we use the *macro-averaging* F1 to evaluate on imbalanced data. Clearly, both evaluation measures require that the ground truth of the sequence sets be known, which is the case in our experiments. We will report the results in the format *average* $\pm$ 1 *standard deviation*.

In the experiments, the primary HMM-based similarity measure was used to construct the *baseline* (implicit) representation to provide context. Specifically, for the sequences s and s', we trained

[1] available at https://www.ncbi.nlm.nih.gov/COG/

their HMMs λ and λ' using the Baum-Welch algorithm [3]; then, the similarity in terms of the mutual likelihoods was computed as $\frac{1}{2}[\log p(s|\lambda') + \log p(s'|\lambda)]$. For PBC, we computed the sample-to-center similarity by averaging the similarities between the test sequence and the training sequences (of the same class).

From the numerous k-gram embedding methods in the literature, we chose DMk [7] for comparisons. It makes use of the k-length overlapping subsequences for the features of the representation model and assigns them entropy-based weights related to their location distribution on the sequence; therefore, the global sequential structure can be extracted by this method to some extent. We used DM2 and DM3 (i.e., DMk with $k = 2$ and 3, respectively) to examine the sensitivity with respect to the gram length.

Two 5-layer encoder-decoder neural networks were also used as competitors in the experiments. The networks are named AENNs (short for AutoEncoder Neural Networks), which are defined with reference to the auto-encoders discussed in [6]. The first network is called LSTM AENN designed for JSI and LIBRAS, as they are originally presented by coordinate series. The remaining four sets were learned by a word-embedding AENN, with the word-embedding layer [16] generating an embedding vector of length 32 for each symbol. The timestep of the inputs was set to the maximal length of the sequences in each set (see Table 1). We implemented the networks using the high-level neural-network API called Keras. The Microsoft Cognitive Toolkit without GPU was used as the backend. The active functions were configured as ReLU, and the Adam optimizer was used for training.

The HMM variant (abbreviated varHMM) [14] was chosen as the representative for the HMM-embedding methods to be compared in the experiments. It represents each sequence by the transition matrix of a shared-state HMM, which is trained by forcing the learning algorithm to compute a unique emission matrix for each state over all the sequences. It can thus be viewed as a special case of our sopHMM without optimization on the topics. The extended Baum-Welch algorithm suggested by the authors was used to train the HMMs, setting its parameters for the Dirichlet prior to 1.

The number of HMM states used in varHMM and the baseline was set to M, the same as the number of symbols. This setting relates to L, the number of initial topics in our sopHMM. On the three human activity sequence sets with relatively small M, the sensitivity to L was evaluated in Appendix B, where we showed that sopHMM can obtain robust results when $L \geq 2 \cdot M$. As $L \geq M$ and sopHMM is $O(L^2)$ in time with respect to L, in the experiments, we set $L = 2 \cdot M$ if $M \leq 10$ and $L = M$ otherwise, to make a trade-off between efficiency and accuracy.

5.3. Human activity recognition

The number of underlying topics (here, the human activities) can be adaptively determined by the hierarchical model selection in our sopHMM. We report here that the numbers of topics were identified as $K = 9$ for the three sequence sets. The accuracies obtained by the different representation methods are summarized in Table 2, where the highest classification quality obtained by each classifier is marked in bold typeface and the results from the baseline are also presented.

It can be seen that, using the representation model generated by our sopHMM, both SVM and PBC can achieve high-quality overall results, being more accurate than the competitors. The results are comparable to the baseline on ADL and LIBRAS, and significantly surpass it on JSI; in fact, the baseline almost fails on JSI with PBC, due to the large difference in the sequence lengths. DMk performs unstably as it is sensitive to the gram length k. The results also show that AENNs, the multi-layer neural network-based auto-encoders, perform moderately. This may be due to under-fitting of the networks, which are trained with small sample sets. We also observe that varHMM is able to achieve better performance than the other competitors in most cases. This is mainly due to the implicit use of the HMM states in representing the "topics" for sequences; however, the states are not optimized like our sopHMM, resulting in a performance gap from the baseline.

To better understand the performance of our sopHMM in symbolic sequence representation, first, we decided to examine the activities identified for the ADLs in more detail. Here, the emission probability distributions of the topics are interpreted as a representation for the human daily activities. Let us take the two confusing activities: Breakfast and Lunch, each of which may involve the same events (symbols) such as Fridge and Microwave. Table 3 summarizes their distributions obtained by varHMM (with the number of states set to 9) and our sopHMM. We see that the topics learned by sopHMM can be more distinguishable due to the use of the optimization method, which makes the states well-scattered and, consequently, provides the model with more capacity to distinguish between sequences.

Next, we show how the representation model describes the structural features within sequences based on the identified topics. The results obtained on LIBRAS were used here because of their intuitiveness. Fig. 3 illustrates three examples in different hand-movement types (classes). Each 3×3 block in the figure stands for a topic and each cell in the block corresponds to one of the symbols (directions). The topic distribution, being a 9-length vector, is illustrated by the brightness of the 9 cells in the block. The arrow indicates the direction associated with the largest probability in the vector.

The state sequences shown in the figure were obtained in three steps. First, the mean vector (as the prototype in PBC) was computed for each class. Then, we selected the sequence with Euclidean distance closest to the prototype as the example; thus, it would be the most representative sequence in the class. Third, the best state (topic) sequence was found using the Viterbi algorithm and the trained HMM optimized by sopHMM. It can be seen from Figure 3 that the representative gestures of Circle, Vertical zigzag, and Face-up curve can be accurately presented by the topic chains (state sequences), where each topic indicates a distinguishable hand-movement mode. Furthermore, the difference between classes can be readily identified according to the distribution of topics in the chain; this is exactly what our new representation model captures (see Definition 1).

5.4. Protein recognition

Due to the large number of symbols and the random initialization of the Baum-Welch reestimation algorithm, the number of topics identified for the protein sets may vary. By examining the 20 results obtained by sopHMM (see Appendix C for details), we found that the number of topics is likely 17 (for the 33 $(\alpha/\beta)_8$-barrel proteins) or 18 (for the COG sets). The classification results yielded by the different methods on the protein sets are summarized in Table 4, where boldface font highlights the highest-accuracy entries. The baseline results are also presented.

The results in Table 4 indicate that the baseline significantly underperforms our sopHMM; in particular, it fails in classifying COG-S2 with PBC. This is due to the large number of HMM states being used; in this case, the difference between sequence likelihoods is likely reduced. The AENN method achieves higher accuracy than varHMM on the protein sequences, especially on the two COG sets. The results demonstrate that neural-network auto-encoders can be more accurate when they are trained with more samples and longer sequences, as discussed in [35].

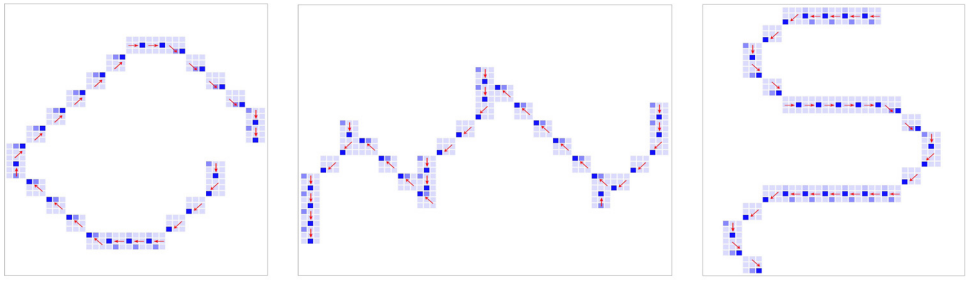
It can be seen from the table that sopHMM is able to achieve significantly high classification accuracies on the protein sequence sets, while outperforming the competitors in most cases. When

Table 2
Classification performance obtained by different representation methods and the baseline on the sequence sets for human activity recognition.

Dataset		sopHMM	varHMM	DM2	DM3	AENN	Baseline
ADL	# features	81	81	64	239	81	-
	CA (SVM)	0.963±0.069	0.867±0.108	0.785±0.138	0.626±0.084	0.724±0.160	0.958±0.075
	F1 (SVM)	0.957±0.081	0.854±0.120	0.772±0.148	0.450±0.138	0.701±0.179	0.954±0.085
	CA (PBC)	0.946±0.076	0.858±0.117	0.737±0.139	0.593±0.147	0.751±0.135	0.966±0.062
	F1 (PBC)	0.938±0.091	0.844±0.128	0.712±0.158	0.513±0.171	0.727±0.159	0.963±0.069
JSI	# features	81	81	80	460	81	-
	CA (SVM)	0.923±0.048	0.851±0.070	0.874±0.066	0.704±0.084	0.588±0.102	0.875±0.067
	F1 (SVM)	0.923±0.049	0.849±0.071	0.872±0.070	0.671±0.107	0.554±0.115	0.873±0.069
	CA (PBC)	0.929±0.052	0.863±0.075	0.878±0.071	0.701±0.086	0.647±0.109	0.525±0.038
	F1 (PBC)	0.929±0.053	0.860±0.080	0.876±0.073	0.666±0.118	0.627±0.118	0.382±0.072
LIBARS	# features	81	81	81	610	81	-
	CA (SVM)	0.900±0.034	0.820±0.047	0.811±0.036	0.689±0.050	0.752±0.046	0.912±0.031
	F1 (SVM)	0.898±0.036	0.817±0.048	0.805±0.036	0.680±0.053	0.739±0.049	0.910±0.032
	CA (PBC)	0.812±0.037	0.732±0.053	0.703±0.043	0.595±0.050	0.594±0.046	0.778±0.036
	F1 (PBC)	0.796±0.041	0.719±0.056	0.689±0.046	0.580±0.053	0.582±0.048	0.749±0.038

Table 3
Comparison of the state probability distributions for the two ADL activities.

Method	Activity	Emission probability distribution								
		Fridge	Microwave	Seat	Basin	Bed	Toilet	Shower	Cupboard	Maindoor
varHMM	Breakfast	**0.6667**	**0.3333**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	Lunch	**0.6724**	0.0000	**0.3276**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
sopHMM	Breakfast	0.0000	**0.8525**	**0.0005**	**0.1466**	0.0000	0.0000	0.0000	0.0000	**0.0004**
	Lunch	**0.8124**	**0.0371**	**0.1505**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000



(a) Circle (b) Vertical zigzag (c) Face-up curve

Fig. 3. Representation of the hand-movement sequences for the three classes in LIBRAS.

Table 4
Comparison of the classification performance on the three protein sets.

Dataset		sopHMM	varHMM	DM2	DM3	AENN	Baseline
33 $(\alpha/\beta)_8$	# features	249	441	404	6168	441	-
-barrel	CA (SVM)	**0.950±0.081**	0.749±0.168	0.934±0.095	0.636±0.185	0.779±0.142	0.894±0.112
	F1 (SVM)	**0.950±0.081**	0.749±0.168	0.934±0.095	0.636±0.185	0.779±0.142	0.894±0.112
	CA (PBC)	**0.930±0.093**	0.744±0.162	0.873±0.114	0.580±0.193	0.741±0.152	0.821±0.133
	F1 (PBC)	**0.930±0.093**	0.744±0.162	0.873±0.114	0.580±0.193	0.741±0.152	0.821±0.133
COG-S1	# features	324	400	400	7868	400	-
	CA (SVM)	0.948±0.014	0.686±0.034	**0.963±0.012**	0.630±0.031	0.815±0.022	0.899±0.018
	F1 (SVM)	0.948±0.014	0.678±0.034	**0.962±0.013**	0.643±0.031	0.816±0.022	0.896±0.019
	CA (PBC)	**0.924±0.018**	0.698±0.033	0.833±0.020	0.578±0.030	0.739±0.025	0.393±0.022
	F1 (PBC)	**0.923±0.018**	0.696±0.032	0.822±0.023	0.605±0.029	0.737±0.026	0.293±0.020
COG-S2	# features	324	400	400	7971	400	-
	CA (SVM)	**0.774±0.024**	0.457±0.028	0.755±0.018	0.411±0.021	0.542±0.020	0.757±0.020
	F1 (SVM)	**0.775±0.024**	0.452±0.028	0.756±0.018	0.413±0.022	0.536±0.020	0.757±0.020
	CA (PBC)	**0.732±0.025**	0.468±0.029	0.623±0.018	0.431±0.019	0.475±0.021	0.050±0.000
	F1 (PBC)	**0.734±0.025**	0.460±0.029	0.609±0.018	0.454±0.021	0.445±0.023	0.050±0.000

2-grams are used to represent the sequences, DMk obtains high-quality results with SVM; however, this is not the case for PBC and when 3-grams are used. As we can see that the number of features proliferates when $k = 3$, resulting in the unnecessarily high-dimensional embedding space of the sequences. With the model selection method used in our sopHMM, however, the number of resulting features can be adaptively determined.

We chose the results obtained on the 33 $(\alpha/\beta)_8$-barrel protein set for further analysis. Applying our sopHMM to this set, each protein was transformed into a 249-dimensional ($17^2 = 249$) vec-

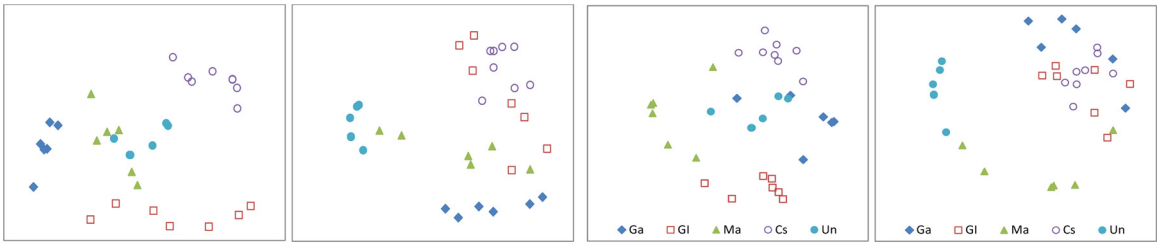


Fig. 4. Distribution of the 33 $(\alpha/\beta)_8$-barrel protein sequences in the vector space.

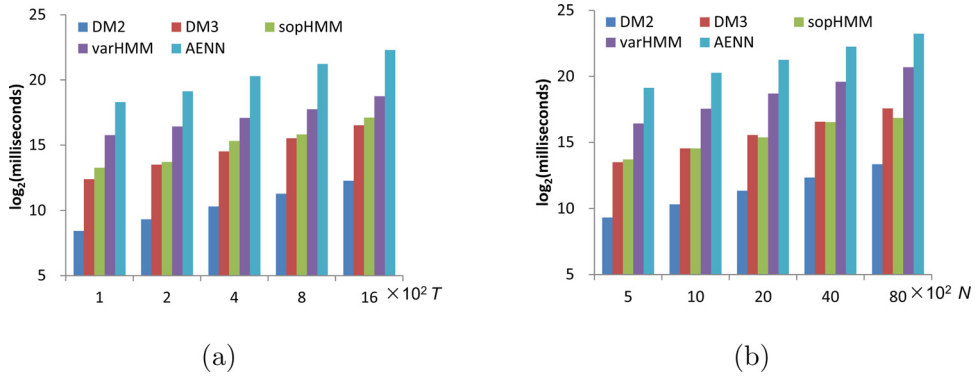


Fig. 5. Relationship of runtime (in logarithmic milliseconds) with different lengths of sequences (a) and different numbers of sequences (b).

tor. Fig. 4 gives a view of the proteins based on the vector representation. We obtained the result by performing principal component analysis (PCA) on the vectors and projecting them onto the 4 principal components, after which each of the 4-dimensional vectors was converted into a unit-length vector. Fig. 4 shows the resulting vectors projected onto some subspaces. It can be seen that the five classes are well grouped in the representation model yielded by sopHMM, resulting in high classification accuracy with both SVM and PBC.

5.5. Scalability

Here, we first examine the scalability of sopHMM with respect to the sequence length (T) and the data size (N). The experiments were conducted on the subsets of the COG database. Five subsets, each containing 500 proteins, were used to test the scalability with respect to T. Proteins with average lengths of 100, 200, 400, 800, and 1600, respectively, were randomly chosen from the COG database for this purpose. To test the scalability with respect to the number of sequences, another five subsets were generated. We randomly chose 500, 1000, 2000, 4000, and 8000 proteins for these subsets, respectively; each protein here has an average length of 200.

Figure 5 illustrates the relationship of the runtime with the length (T) and the number of sequences (N). The runtime is shown in logarithmic milliseconds for ease of comparison. We can see that the runtime of sopHMM increases sublinearly with respect to T and N. The fastest method is DM2, while the neural-network autoencoder is much slower than the others. For example, on the proteins set with $T = 200$ and $N = 4000$, AENN consumed about 4954 seconds, while the number was 94.6 using our sopHMM.

The efficiency of sopHMM is between DM3 and varHMM, and is much more efficient than varHMM, which is also an HMM-based method. Since varHMM trains each sequence by sharing a unique set of states, it takes more time to converge. In our sopHMM, each sequence is initially trained with its own states, and the shared states are learned in a tree-based model selection process; therefore, it is more efficient than varHMM.

6. Conclusion and perspectives

We have proposed a new HMM-based method to embed symbolic sequences in a real space. We derived the new representation based on the assumption that sequences in the same set are originated from a unique K-topic generative model, where each topic presents one discriminating structural feature shared by all sequences in the set. We modeled the underlying topics by the a set of well-scattered HMM states, allowing sequences to be explicitly represented by the HMM parameters indicating their respective distribution of topics in the chain.

We have used the transition probability distribution of topics for the representation, defined from the perspective of Bhattacharyya kernel and can be obtained in an unsupervised way. These features enable many pattern recognition algorithms originally developed for vector data, more than the SVM and PBC used in the experiments, to be applied to symbolic sequences. As the matrix representation might involve redundant information, in the future, we shall introduce other probability kernels and matrix decomposition techniques, such as singular value decomposition (SVD), to derive a more compact representation model.

Moreover, we transformed the problem of learning the discriminating topics into a new problem of Markovian state optimization, for which an efficient method call sopHMM was proposed. Here, the topics are first obtained by HMM states clustering, which aims to optimize the topic homogeneity, and then they are refined in the hierarchical model selection procedure according to the average topic-scatter that measures topic distinguishability. This method has a feature that allows the discrimination quality of the states to be optimized while determining the optimal number of states adaptively. The existing HMM optimization methods lack this feature.

We applied sopHMM to represent symbolic sequences for human activity recognition and protein recognition in the experiments, and observed that the resulting topics can be explainable and distinguishable, as the topic distribution is usually concentrated on a few key symbols. In effect, the learning process is equivalent to performing a built-in sequence feature (symbol or

event) selection. This property will be useful for other sequence-based recognition tasks, e.g., finding sequential patterns from large sequence data.

In the present work, sequences require re-training to obtain the representation model in Step 3 of Algorithm 1. Despite its efficiency, an interesting future exploration would be to learn the topics and the representation in a joint optimization procedure. Our future efforts will also be directed toward developing supervised techniques to learn the K-topic model.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

Lifei Chen and Haiyan Wu are the co-first authors of this paper. This work was supported by the National Natural Science Foundation of China under Grant Nos. U1805263, 61976053, 61672157, the Foundation of Fujian Provincial Department of Science and Technology under Grant Nos. 2019J01427, 2020H0011, 2019H0009.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.patcog.2022.108849.

References

[1] L. Dang, K. Min, H. Wang, M. Piran, C. Lee, H. Moon, Sensor-based and vision-based human activity recognition: a comprehensive survey, Pattern Recognit. 108 (2020) 107561.
[2] Q. Zou, G. Lin, X. Jiang, X. Liu, X. Zeng, Sequence clustering in bioinformatics: an empirical study, Brief. Bioinform.s 21 (2020) 1–10.
[3] G. Fink, Markov Models for Pattern Recognition: From Theory to Applications, Springer-Verlag, Berlin Heidelberg, 2008.
[4] M. Bicego, E. Pekalska, D. Tax, R. Duin, Component-based discriminative classification for hidden markov models, Pattern Recognit. 42 (11) (2009) 2637–2648.
[5] R. Tillquist, M. Lladser, Low-dimensional representation of genomic sequences, J. Math. Biol. 79 (2019) 1–29.
[6] N. Srivastava, E. Mansimov, R. Salakhutdinov, Unsupervised learning of video representations using LSTMs, in: In: Proceedings of the International Conference on Machine Learning, 2015, pp. 843–852.
[7] D. Wei, Q. Jiang, Y. Wei, S. Wang, A novel hierarchical clustering algorithm for gene sequences, BMC Bioinform. 13 (2012) 174.
[8] L. Yuan, W. Wang, L. Chen, Two-stage pruning method for gram-based categorical sequence clustering, Int. J. Mach. Learn. Cyb. 10 (4) (2019) 631–640.
[9] G. Guo, L. Chen, Y. Ye, Q. Jiang, Cluster validation method for determining the number of clusters in categorical sequences, IEEE Trans. Neural Netw. Learn. Syst. 28 (12) (2017) 2936–2948.
[10] X. Gao, J. Zhang, Z. Wei, Deep learning for sequence pattern recognition, in: Proceedings of the IEEE International Conference on Networking, Sensing and Control, 2018, pp. 1–6.
[11] H. Shi, C. Zhang, Q. Yao, Y. Li, F. Sun, D. Jin, State-sharing sparse hidden Markov model for personalized sequences, in: Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2019, pp. 1549–1559.
[12] K. Huang, X. Fu, N. Sidiropoulos, Learning hidden Markov models from pairwise Co-occurrences with application to topic modeling, in: Proceedings of the International Conference on Machine Learning, 2018, pp. 2068–2077.
[13] J. Vaičiulytė, L. Sakalauskas, Recursive estimation of multivariate hidden Markov model parameters, Comput. Stat. 34 (2019) 1337–1353.
[14] S. Blasiak, H. Rangwala, A hidden Markov model variant for sequence classification, in: Proceedings of the International Joint Conference on Artificial Intelligence, 2011, pp. 1192–1197.
[15] S. Kiefer, A. Sistla, Distinguishing hidden Markov chains, in: Proceedings of the Symposium on Logic in Computer Science, 2016, pp. 66–75.
[16] C. Boom, S. Canneyt, T. Demeester, B. Dhoedt, Representation learning for very short texts using weighted word embedding aggregation, Pattern Recognit. Lett. 80 (1) (2016) 150–156.
[17] D. Martin, Distributions of pattern statistics in sparse Markov models, Ann. I. Stat. Math. 72 (4) (2020) 895–913.
[18] J. Pylkkönen, K. M, Analysis of extended baumwelch and constrained optimization for discriminative training of HMMs, IEEE Trans. Audio Speech Lang. Process. 20 (9) (2012) 2409–2419.
[19] F. Qin, A. Auerbach, F. Sachs, A direct optimization approach to hidden Markov modeling for single channel kinetic, Biophys. J. 79 (2000) 1915–1927.
[20] A. Emdadi, F. Moughari, F. Meybodi, C. Eslahchi, A novel algorithm for parameter estimation of hidden Markov model inspired by ant colony optimization, Heliyon, 5 (2019) e01299.
[21] S. Günter, H. Bunke, Optimizing the number of states, training iterations and Gaussians in an HMM-based handwritten word recognizer, in: Proceedings of the International Conference on Document Analysis and Recognition, 2003, pp. 472–476.
[22] G. Rigoll, J. Geiger, J. Schenk, F. Wallhoff, Optimizing the number of states for HMM-based on-line handwritten whiteboard recognition, in: Proceedings of the International Conference on Frontiers in Handwriting Recognition, 2010, pp. 107–112.
[23] M. Dua, R. Aggarwal, M. Biswas, Discriminative training using noise robust integrated features and refined HMM modeling, J. Intell. Syst. 29 (1) (2020) 327–344.
[24] M. Bicego, V. Murino, Investigating hidden Markov models capabilities in 2D shape classification, IEEE Trans. Pattern Anal. Mach. Intell. 26 (2) (2004) 281–286.
[25] S. Chatzis, D. Kosmopoulos, A variational Bayesian methodology for hidden Markov models utilizing student's-t mixtures, Pattern Recognit. 44 (2) (2011) 295–306.
[26] J. Jebara, R. Kondor, A. Howard, Probability product kernels, J. Mach. Learn. Res. 5 (2004) 819–844.
[27] J. Chung, P. Kannappan, C. Ng, P. Sahoo, Measures of distance between probability distributions, J. Math. Anal. Appl. 138 (1) (1989) 280–292.
[28] J. Goldberger, S. Roweis, Hierarchical clustering of a mixture model, in: Proceedings of the International Conference on Neural Information Processing Systems, 2004, pp. 505–512.
[29] E. Coviello, A. Chan, G. Lanckriet, Clustering hidden Markov models with variational HEM, J. Mach. Learn. Res. 15 (2) (2014) 697–747.
[30] F. Ordóñez, P. de Toledo, A. Sanchis, Activity recognition using hybrid generative/discriminative models on home environments using binary sensors, Sensors 13 (2013) 5460–5477.
[31] B. Kaluza, V. Mirchevska, E. Dovgan, M. Lustrek, M. Gams, An agent-based approach to care in independent living, in: Proceedings of the International Joint Conference on Ambient Intelligence, 2010, pp. 177–186.
[32] D. Dias, R. Madeo, T. Rocha, H. Biscaro, S.M. Peres, Hand movement recognition for brazilian sign language: a study using distance-based neural networks, in: Proceedings of the International Joint Conference on Neural Networks, 2009, pp. 697–704.
[33] C. Burges, A tutorial on support vector machines for pattern recognition, Data Min. Knowl. Discov. 2 (1998) 121–167.
[34] H. Takçı, T. Güngör, A high performance centroid-based classification approach for language identification, Pattern Recognit. Lett. 33 (16) (2012) 2077–2084.
[35] S. Makridakis, E. Spiliotis, V. Assimakopoulos, Statistical and machine learning forecasting methods: concerns and ways forward, PLoS ONE 13 (3) (2018) e0194889.

Lifei Chen received his bachelor's degree in computer science from University of Electronic Science and Technology of China in 1993; his MSE from Tsinghua University, China, in 2004; and his Ph.D. from Xiamen University, China, in 2008. He is a professor at the Department of Computer Science, Fujian Normal University, China. His research interests include machine learning and pattern recognition.

Haiyan Wu received her bachelor's degree and MS in computer science from Fuzhou University, China. She is now a Ph.D. student at Fujian Normal University, China. Her research interests include statistical machine learning and pattern recognition.

Wenxuan Kang received his bachelor's degree in computer science from Fujian Normal University, China. He is currently a MS student at Fujian Normal University. His research interests include machine learning and data mining.

Shengrui Wang received his bachelor's degree from Hebei University, his MS from the Université Joseph Fourier, and his Ph.D. from the Institut National Polytechnique de Grenoble. He is a full professor at Université de Sherbrooke, Canada. His research interests include pattern recognition, machine learning, data mining, and bioinformatics.