Artificial Intelligence 246 (2017) 53–85

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Hierarchical semi-Markov conditional random fields for deep recursive sequential data

Truyen Tran [a,*], Dinh Phung [a], Hung Bui [b], Svetha Venkatesh [a]

[a] Center for Pattern Recognition and Data Analytics, Deakin University Geelong, Australia
[b] Adobe Research, Adobe, USA

ARTICLE INFO

Article history:
Received 20 January 2015
Received in revised form 12 February 2017
Accepted 14 February 2017
Available online 24 February 2017

Keywords:
Deep nested sequential processes
Hierarchical semi-Markov conditional random field
Partial labelling
Constrained inference
Numerical scaling

ABSTRACT

We present the *hierarchical semi-Markov conditional random field* (HSCRF), a generalisation of linear-chain conditional random fields to model deep nested Markov processes. It is parameterised as a conditional log-linear model and has polynomial time algorithms for learning and inference. We derive algorithms for partially-supervised learning and constrained inference. We develop numerical scaling procedures that handle the overflow problem. We show that when depth is two, the HSCRF can be reduced to the semi-Markov conditional random fields. Finally, we demonstrate the HSCRF on two applications: (i) recognising human activities of daily living (ADLs) from indoor surveillance cameras, and (ii) noun-phrase chunking. The HSCRF is capable of learning rich hierarchical models with reasonable accuracy in both fully and partially observed data cases.

1. Introduction

Modelling hierarchical depth in complex stochastic processes is important in many application domains. In a deep hierarchy, each level is an *abstraction* of lower level details [1–4]. This paper studies *recursively sequential* processes, in that each level is a sequence and each node in a sequence can be decomposed further into a sub-sequence of finer grain [2].

Consider, for example, a frequent activity performed by human 'eat-breakfast'. It may include a series of more specific activities like 'enter-kitchen', 'go-to-cupboard', 'take-cereal', 'wash-dishes' and 'leave-kitchen'. Each specific activity can be decomposed into finer details. Similarly, in natural language processing (NLP) syntax trees are inherently hierarchical. In a partial parsing task known as noun-phrase (NP) chunking [5], there are three syntactic levels: the sentence, noun-phrases and part-of-speech (POS) tags. In this setting, the sentence is a sequence of NPs and non-NPs, and each phrase is a sub-sequence of POS tags.

A popular approach to deal with hierarchical data is to build a *cascaded* model where each level is modelled separately, and the output of the lower level is used as the input of the level right above it (e.g. see [6]). For instance, in NP chunking this approach first builds a POS tagger and then constructs a chunker that incorporates the output of the tagger. This approach is sub-optimal because the POS tagger takes no information of the NPs and the chunker is not aware of the reasoning of the tagger. In contrast, a noun-phrase is often very informative to infer the POS tags belonging to the phrase. As a result, this layered approach may suffer from the so-called *cascading error* problem in that errors introduced from the lower layer propagate to higher levels.

* Corresponding author.
E-mail address: truyen.tran@deakin.edu.au (T. Tran).

http://dx.doi.org/10.1016/j.artint.2017.02.003
0004-3702/

A more holistic approach is to build a joint representation of all the levels. Formally, given a data sequence z we need to model and infer about the deep, nested semantic x. The main problem is to choose an appropriate representation of x so that inference can be efficient. An important class of representation is hierarchical hidden Markov model (HHMM) [2]. An HHMM is a nested hidden Markov network (HMM) in the sense that each state is also a sub HMM. Although HMMs represent only first-order Markov processes, HHMMs offer higher-order interaction. HHMMs are generative models with joint distribution $\Pr(x, z)$, where the data generating distribution $\Pr(z \mid x)$ must be simplified for efficient inference about the semantic $\Pr(x \mid z)$. An alternative is to model the discriminative distribution $\Pr(x \mid z)$ directly without modelling the data $\Pr(z)$. This can be more effective since arbitrary long-range and interdependent data features can be incorporated into the model.

The most popular class of probabilistic structured output methods are conditional random fields (CRFs) [7], but the early models are flat. Deep variants have been introduced the past decade, including dynamic CRFs (DCRF) [8], hierarchical CRFs [9,10]), and stacked CRFs [11]. However, these methods require a fixed pre-defined hierarchy, and thus are not suitable for problems with automatically inferred topologies.

To this end, we construct a novel discriminative model called *Hierarchical Semi-Markov Conditional Random Field* (HSCRF).[1] The HSCRF offers nested semantic similar to that by the HHMM but is parameterised as an undirected log-linear model. The HSCRF generalises linear-chain CRFs [7] and semi-Markov CRFs [13].

To be more concrete, let us return to the NP chunking example. The problem can be modelled as a three-level HSCRF, where the root represents the sentence, the second level the NP process, and the bottom level the POS process. The root and the two processes are conditioned on the sequence of words in the sentence. Under discriminative modelling, rich contextual information can be simply encoded as features including starting and ending of a phrase, phrase length, and distribution of words falling inside the phrase can be effectively encoded. On the other hand, such encoding is much more difficult for HHMMs.

We then proceed to address important issues. First, we show how to represent HSCRFs using a dynamic graphical model (e.g. see [14]) which effectively encodes hierarchical and temporal semantics. For parameter learning, an efficient algorithm based on the Asymmetric Inside–Outside of [15] is introduced. For inference, we generalise the Viterbi algorithm to decode the semantics from an observational sequence.

The common assumptions in discriminative learning and inference are that the training data in learning is fully labelled, and the test data during inference is not labelled. We propose to relax these assumptions in that training labels may only be partially available. Likewise, when some labels are given during inference, the algorithm should automatically adjust to meet the new constraints.

We demonstrate the effectiveness of HSCRFs in two applications: (i) segmenting and labelling activities of daily living (ADLs) in an indoor environment and (ii) jointly modelling noun-phrases and part-of-speeches in shallow parsing. Our experimental results in the first application show that the HSCRFs are capable of learning rich, hierarchical activities with good accuracy and exhibit better performance when compared to DCRFs and flat-CRFs. Results for the partially supervised case also demonstrate that significant reduction of training labels still results in models that perform reasonably well. We also show that observing a small amount of labels can significantly increase the accuracy during decoding. In shallow parsing, the HSCRFs can achieve higher accuracy than standard CRF-based techniques and the recent DCRFs.

To summarise, in this paper we claim the following contributions:

- Introducing a novel Hierarchical Semi-Markov Conditional Random Field (HSCRF) to model complex hierarchical and nested Markovian processes in a discriminative framework.
- Developing an efficient generalised Asymmetric Inside–Outside (AIO) algorithm for full-supervised learning.
- Generalising the Viterbi algorithm for decoding the most probable semantic labels and structure given an observational sequence.
- Addressing the problem of partially-supervised learning and constrained inference.
- Constructing a numerical scaling algorithm to prevent numerical overflow.
- Demonstration of the applicability of the HSCRFs for modelling human activities in the domain of home video surveillance and shallow parsing of English.

The rest of the paper is organised as follows. Section 2 reviews Conditional Random Fields and Hierarchical Hidden Markov Models. Section 3 continues with the HSCRF model definition. Section 4 defines building blocks required for common inference tasks. Section 5 presents the generalised Viterbi algorithm. Parameterisation and estimation follow in Section 6. Learning and inference with partially available labels are addressed in Section 7. Section 8 presents a method for numerical scaling to prevent numerical overflow. Section 9 documents experimental results. Section 11 concludes the paper.

[1] Preliminary version was published in NIPS'08 [12].

Table 1
Notations used in this paper.

Notation	Description
$x_{i:j}^{d:d'}$	Subset of state variables from level d down to level d' and starting from time i and ending at time j, inclusive
$e_{i:j}^{d:d'}$	Subset of ending indicators from level d down to level d' and starting from time i and ending at time j, inclusive
$\zeta_{i:j}^{d,s}$	Set of state variables and ending indicators of a sub model rooted at s^d, level d, spanning a sub-string $[i, j]$
c	Contextual clique
i, j, t	Time indices
τ^d	Set of all ending time indices, e.g. if $i \in \tau^d$ then $e_i^d = 1$
r, s, u, v, w	State
$R_{i:j}^{d,s,z}$	State-persistence potential of state s, level d, spanning $[i, j]$
$\pi_{u,i}^{d,s}$	Initialisation potential of state s at level d, time i initialising sub-state u
$A_{u,v,i}^{d,s,z}$	Transition at level d, time i from state u to v under the same parent s
$E_{u,i}^{d,s,z}$	Ending potential of state z at level d and time i, and receiving the return control from the child u
$\Phi[\zeta, z]$	The global potential of a particular configuration ζ given observation sequence z
S^d	The set of state symbols at level d
$\Delta_{i:j}^{d,s}$	The symmetric inside mass for state s at level d, spanning substring $[i, j]$
$\hat{\Delta}_{i:j}^{d,s}$	The full symmetric inside mass for state s at level d, spanning substring $[i, j]$
$\Lambda_{i:j}^{d,s}$	The symmetric outside mass for state s at level d, spanning substring $[i, j]$
$\hat{\Lambda}_{i:j}^{d,s}$	The full symmetric outside mass for state s at level d, spanning substring $[i, j]$
$\alpha_{i:j}^{d,s}(u)$	The asymmetric inside mass for parent state s at level d, starting at i and having a child-state u which returns control to the parent or transits to a new child-state at j
$\lambda_{i:j}^{d,s}(u)$	The asymmetric outside mass, as a counterpart of asymmetric inside mass $\alpha_{i:j}^{d,s}(u)$
$\delta[\cdot], \mathbb{I}[\cdot]$	Indicator functions
$\psi(\cdot),\varphi(\cdot)$	Potential functions.

2. Preliminaries

This section presents foundations upon which the proposed HSCRF is built: conditional random fields and hierarchical hidden Markov models. For later reference, we define mathematical notations in Table 1.

2.1. Conditional random fields

Denote by $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ the graph where $\mathscr{V}$ is the set of vertices, and $\mathscr{E}$ is the set of edges. Associated with each vertex i is a state variable x_i Let x be joint state variable, i.e. $x = (x_i)_{i \in \mathscr{V}}$. *Conditional random fields* (CRFs) [7] define a conditional distribution given the observation z as follows

$$\Pr(x \mid z) = \frac{1}{Z(z)} \prod_c \phi_c(x_c, z) \tag{1}$$

where c is the index of cliques in the graph, $\phi_c(x_c, z)$ is a non-negative potential function defined over the clique c, and $Z(z) = \sum_x \prod_c \phi_c(x_c, z)$ is the partition function.

Let $\{\tilde{x}\}$ be the set of observed state variables with the empirical distribution $Q(\tilde{x})$, and $\mathbf{w}$ be the parameter vector. Learning in CRFs is typically by maximising the (log) likelihood

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \mathscr{L}(\mathbf{w}) = \arg\max_{\mathbf{w}} \sum_{\tilde{x}} Q(\tilde{x}) \log \Pr(\tilde{x} \mid z; \mathbf{w}) \tag{2}$$

The gradient of the log-likelihood can be computed as

$$\nabla \mathscr{L}(\mathbf{w}) = \sum_{\tilde{x}} Q(\tilde{x}) \sum_c \left(\nabla \log \phi_c(\tilde{x}_c, z) - \sum_{x_c} \Pr(x_c \mid z) \nabla \log \phi_c(x_c, z) \right) \tag{3}$$

Thus, the inference needed in CRF parameter estimation is the computation of clique marginals $\Pr(x_c|z)$.

Typically, CRFs are parameterised as log-linear models, i.e. $\phi_c(x_c, z) = \exp(\mathbf{w}^\top \mathbf{f}(x_c, z))$, where $\mathbf{f}(.)$ is the feature vector and $\mathbf{w}$ is weight vector. Let $\mathbf{F}(x, z) = \sum_c \mathbf{f}(x_c, z)$ be the global feature. Eq. (3) can be written as follows

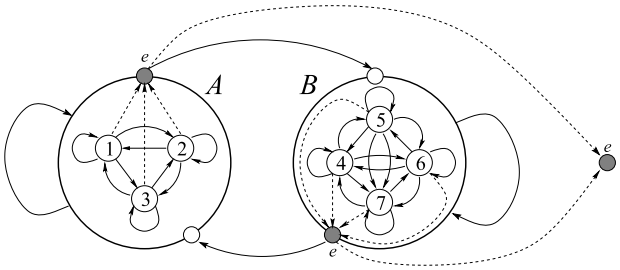

Fig. 1. The state transition diagram of an HHMM.

$$\nabla \mathscr{L} = \sum_{\tilde{x}} Q(\tilde{x}) \sum_{c} \left(\mathbf{f}(\tilde{x}_c, z) - \sum_{x_c} \Pr(x_c \mid z) \mathbf{f}(x_c, z) \right) \tag{4}$$

$$= \mathbb{E}_{Q(\tilde{x})}[\mathbf{F}] - \mathbb{E}_{\Pr(x|z)}[\mathbf{F}] \tag{5}$$

Thus gradient-based maximum likelihood learning in the log-linear setting boils down to estimating the feature expectations, also known as expected sufficient statistics (ESS).

2.1.1. Learning with partial labels

Let $\tilde{x} = (v, h)$, where v is the set of visible variables, and h is the set of hidden variables. The incomplete log-likelihood and its gradient are given as

$$\mathscr{L} = \sum_{\tilde{x}} Q(\tilde{x}) \log \Pr(v \mid z) = \sum_{\tilde{x}} Q(\tilde{x}) \log \sum_{h} \Pr(v, h \mid z)$$

$$= \sum_{\tilde{x}} Q(\tilde{x}) (\log Z(v, z) - \log Z(z)) \tag{6}$$

where $Z(v, z) = \sum_h \prod_c \phi_c(v_c, h_c, z)$. The gradient reads

$$\nabla \mathscr{L} = \mathbb{E}_{h|v,z}[\mathbf{F}(v, h, z)] - \mathbb{E}_{x|z}[\mathbf{F}(x, z)]$$

$$= \sum_{\tilde{x}} Q(\tilde{x}) \sum_{c} \left(\sum_{h_c} \Pr(h_c \mid v, z) \mathbf{f}(v_c, h_c, z) - \sum_{x_c} \Pr(x_c \mid z) \mathbf{f}(x_c, z) \right) \tag{7}$$

2.1.2. Sequential models

The most popular form of CRFs is linear-chain of order n, where n is typically a small integer. This allows fast estimation of the clique marginals $\Pr(x_c \mid z)$ using a forward-backward procedure with time complexity of $\mathcal{O}\left(TK^{n+1}\right)$ for sequence length T and K states.

A generalisation of chain-CRF is semi-Markov CRF (SemiCRF) [13], which is first-order Markovian in segments (but non-Markovian in states). A forward-backward procedure is adapted accordingly with time complexity of $\mathcal{O}\left(TLK^2\right)$ where L is the maximum segment length. In Appendix C we will show that the SemiCRF is a special case of the proposed HSCRF.

2.2. Hierarchical hidden Markov models

Hierarchical HMMs are generalisation of HMMs [16] in that a state in an HHMM may be a sub-HHMM. Thus, an HHMM is a nested Markov chain. In the temporal evolution of HHMM, when a child Markov chain terminates, it returns the control to its parent. Nothing from the terminated child chain is carried forward. Thus, the parent state abstracts out everything belonging to it. Upon receiving the return control the parent then either transits to a new parent or terminates.

Fig. 1 illustrates the state transition diagram of a two-level HHMM. At the top level there are two parent states $\{A, B\}$. Parent A has three children, i.e. $ch(A) = \{1, 2, 3\}$ and B has four, i.e. $ch(B) = \{4, 5, 6, 7\}$. At the top level the transitions are between A and B, as in a normal directed Markov chain. Under each parent there are also transitions between child states, which only depend on the direct parent (either A or B). There are special ending states (represented as shaded nodes in Fig. 1) to signify the termination of the Markov chains. At each time step of the child Markov chain, a child will emit an observational symbol (not shown here).

The temporal evolution of the HHMM can be represented as a dynamic Bayesian network (DBN), which was first done in [17]. Fig. 2 depicts a DBN structure of 3 levels. Associated with each state is an ending indicator to signify the termination of the state. Denote by x_t^d and e_t^d the state and ending indicator at level d and time t, respectively. When $e_t^d = 0$, the state x_t^d continues, i.e. $x_t^d = x_{t+1}^d$. And when $e_t^d = 1$, the state x_t^d transits to a new state, or transits to itself. There are hierarchical

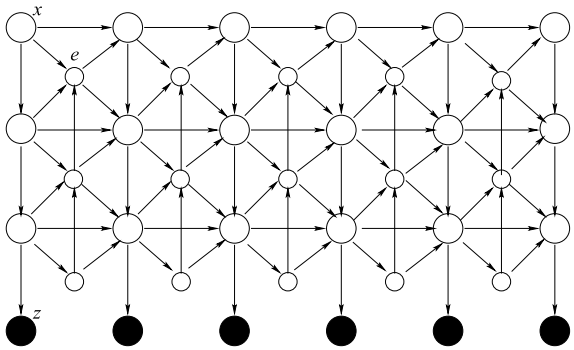

Fig. 2. Dynamic Bayesian network representation of HHMMs.

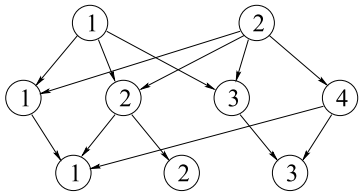

Fig. 3. The shared topological structure.

consistency rules that must be strictly observed. Whenever a state persists (i.e. $e_t^d = 0$), all of the states above it must also persist (i.e. $e_t^{d'} = 0$ for all $d' < d$). Similarly, whenever a state ends (i.e $e_t^d = 1$), all of the states below it must also end (i.e. $e_t^{d'} = 1$ for all $d' > d$).

Inference and learning in HHMMs follow the Inside–Outside algorithm of the probabilistic context-free grammars. Overall, the algorithm has $\mathcal{O}(K^3DT^3)$ time complexity where K is the maximum size of the state space at each level, D is the depth of the model and T is the model length.

When representing as a DBN, the whole stack of states $x_t^{1:D}$ can be collapsed into a 'mega-state' of a big HMM, and therefore inference can be carried out in $\mathcal{O}(K^{2D}T)$ time. This is efficient for a shallow model (i.e. D is small), but problematic for a deep model (i.e. D is large).

3. Model definition

In this section we define the general HSCRF as a hierarchically nested Markov process. Specific log-linear parameterisation will be presented in Sec. 6.1. In an HSCRF, like its generative counterpart (HHMM, Sec. 2.2), each parent state embeds a child Markov chain whose states may in turn contain child Markov chains. The family relation is defined in a *topology*, which is a state hierarchy of depth $D > 1$. The model has a set of states S^d at each level $d \in [1, D]$, i.e. $S^d = \{1...K^d\}$. For each state $s^d \in S^d$ where $1 \leq d < D$, the topological structure also defines a set of children $ch(s^d) \subset S^{d+1}$. Conversely, each child s^{d+1} has a set of parents $pa(s^{d+1}) \subset S^d$. Unlike the original HHMMs where the child states belong exclusively to the parent, the HSCRFs allow arbitrary sharing of children between parents. For example, in Fig. 3, $ch(s^1 = 1) = \{1, 2, 3\}$, and $pa(s^3 = 1) = \{1, 2, 4\}$. This helps avoid an explosive number of sub-states when D is large, leading to fewer parameters and possibly less training data and time. The shared topology has been investigated in the context of HHMMs in [15].

The temporal evolution in the nested Markov processes with sequence length of T operates as follows:

- As soon as a state is created at level $d < D$, it *initialises* a child state at level $d+1$. The initialisation continues downward until reaching the bottom level.
- As soon as a child process at level $d + 1$ *ends*, it returns control to its parent at level d, and in the case of $d > 1$, the parent either *transits* to a new parent state or returns to the grand-parent at level $d - 1$.

In hierarchical nesting, the life-span of a child process belongs exclusively to the life-span of its parent. For example, a parent process at level d starts a new state $s_{i:j}^d$ at time i and persists until time j. At time i the parent initialises a child state s_i^{d+1} which continues until it ends at time $k < j$, at which the child state transits to a new child state s_{k+1}^{d+1}. The child process exits at time j, returning the control to the parent $s_{i:j}^d$. Upon receiving the control the parent state $s_{i:j}^d$ may transit to a new parent state $s_{j+1:l}^d$, or end at j, returning the control to the grand-parent at level $d - 1$.

We now formally specify the nested Markov processes. Let us introduce a multi-level temporal graphical model of length T with D levels, starting from the top as 1 and the bottom as D (Fig. 4). At each level $d \in [1, D]$ and time index $i \in [1, T]$,

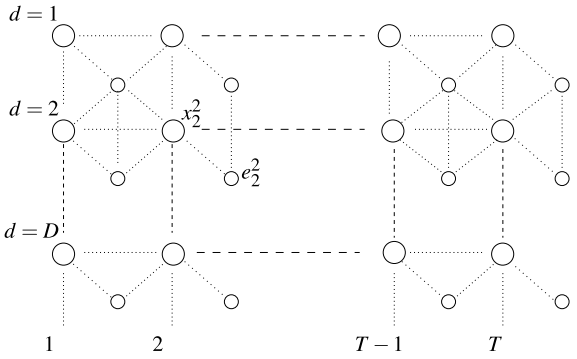

Fig. 4. The multi-level temporal model. Adapted from [18].

Table 2
Hierarchical constraints.

- The top state persists during the course of evolution, i.e. $e_{1:T-1}^1 = 0$.
- When a state finishes, all of its descendants must also finish, i.e. $e_i^d = 1$ implies $e_i^{d+1:D} = 1$.
- When a state persists, all of its ancestors must also persist, i.e. $e_i^d = 0$ implies $e_i^{1:d-1} = 0$.
- When a state transits, its parent must remain unchanged, i.e. $e_i^d = 1$, $e_i^{d-1} = 0$.
- The bottom states do not persists, i.e. $e_i^D = 1$ for all $i \in [1, T]$.
- All states end at T, i.e. $e_T^{1:D} = 1$.

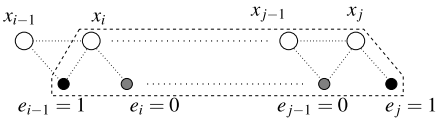

Fig. 5. An example of a state-persistence sub-graph. Adapted from [18].

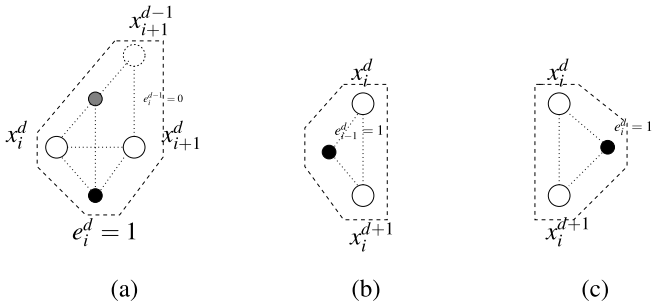

Fig. 6. Sub-graphs for state transition (a), initialisation (b) and ending (c). Adapted from [18].

there is a node representing a state variable $x_i^d \in S^d = \{1, 2, ..., K^d\}$. Associated with each x_i^d is an ending indicator e_i^d signifying whether the state x_i^d ends or persists at i. The nesting nature of the HSCRFs is now realised by imposing the specific constraints on the value assignment of ending indicators as summarised in Table 2.

Thus, specific value assignments of ending indicators provide *contexts* that realise the evolution of the model states in both hierarchical (vertical) and temporal (horizontal) directions. Each context at a level and associated state variables form a *contextual clique*, and we identify four contextual clique types:

- *State-persistence*: This corresponds to the life time of a state at a given level (see Fig. 5). Specifically, given a context $c = \left[e_{i-1:j}^d = (1, 0, .., 0, 1)\right]$, then $\sigma_{i:j}^{persist,d} = \left(x_{i:j}^d, c\right)$, is a contextual clique that specifies the life-span $[i, j]$ of any state $s = x_{i:j}^d$.
- *State-transition*: This corresponds to a state at level $d \in [2, D]$ at time i transiting to a new state (see Fig. 6a). Specifically, given a context $c = \left[e_i^{d-1} = 0, e_i^d = 1\right]$ then $\sigma_i^{transit,d} = \left(x_{i+1}^{d-1}, x_{i:i+1}^d, c\right)$ is a contextual clique that specifies the transition of x_i^d to x_{i+1}^d at time i under the same parent x_{i+1}^{d-1}.

Table 3
Short-hands for contextual clique potentials.

- $R^{d,s,z}_{i:j} = \psi\left(\sigma^{persist,d}_{i:j}, z\right)$ where $s = x^d_{i:j}$.
- $A^{d,s,z}_{u,v,i} = \psi\left(\sigma^{transit,d}_{i}, z\right)$ where $s = x^{d-1}_{i+1}$ and $u = x^d_i$, $v = x^d_{i+1}$.
- $\pi^{d,s,z}_{u,i} = \psi\left(\sigma^{init,d}_{i}, z\right)$ where $s = x^d_i$, $u = x^{d+1}_i$.
- $E^{d,s,z}_{u,i} = \psi\left(\sigma^{end,d}_{i}, z\right)$ where $s = x^d_i$, $u = x^{d+1}_i$.

- *State-initialisation*: This corresponds to a state at level $d \in [1, D-1]$ initialising a new child state at level $d+1$ at time i (see Fig. 6b). Specifically, given a context $c = \left[e^d_{i-1} = 1\right]$, then $\sigma^{init,d}_i = \left(x^d_i, x^{d+1}_i, c\right)$ is a contextual clique that specifies the initialisation at time i from the parent x^d_i to the child x^{d+1}_i.
- *State-ending*: This corresponds to a state at level $d \in [1, D-1]$ ending at time i (see Fig. 6c). Specifically, given a context $c = \left[e^d_i = 1\right]$, then $\sigma^{end,d}_i = \left(x^d_i, x^{d+1}_i, c\right)$ is a contextual clique that specifies the ending of x^d_i at time i with the last child x^{d+1}_i.

In the HSCRF we are interested in the *conditional* setting in which the entire state variables and ending indicators $\left(x^{1:D}_{1:T}, e^{1:D}_{1:T}\right)$ are conditioned on an observational sequence z. For example, in NLP the observation is a sequence of words and the state variables might be the part-of-speech tags and the phrases.

To capture the correlation between variables and such conditioning, we define a positive potential function $\psi(\sigma, z)$ over each contextual clique σ. Table 3 shows the notations for potentials that correspond to the four contextual clique types we have identified above. Details of potential specification are described in the Sec. 6.1.

Let $\zeta = \left(x^{1:D}_{1:T}, e^{1:D}_{1:T}\right)$ denote the set of all variables that satisfies the set of hierarchical constraints listed in Table 2. Let τ^d denote the ordered set of all ending time indices at level d, i.e. if $i \in \tau^d$ then $e^d_i = 1$. The joint potential defined for each configuration is the product of all contextual clique potentials over all ending time indices $i \in [1, T]$ and all semantic levels $d \in [1, D]$:

$$\Phi[\zeta, z] = \left[\prod_{d\in[1,D]} \prod_{i_k, i_{k+1} \in \tau^d} R^{d,s,z}_{i_k+1:i_{k+1}}\right] \prod_{d\in[1,D-1]} \left\{\left[\prod_{i_k \in \tau^{d+1}, i_k \notin \tau^d} A^{d+1,s,z}_{u,v,i_k}\right]\left[\prod_{i_k \in \tau^{d+1}} \pi^{d,s,z}_{u,i_k+1}\right]\left[\prod_{i_k \in \tau^{d+1}} E^{d,s,z}_{u,i_k}\right]\right\} \tag{8}$$

The conditional distribution is given as

$$\Pr(\zeta|z) = \frac{1}{Z(z)} \Phi[\zeta, z] \tag{9}$$

where $Z(z) = \sum_\zeta \Phi[\zeta, z]$ is the partition function for normalisation.

In what follows we omit z for clarity, and implicitly use it as part of the partition function Z and the potential $\Phi[.]$. It should be noted that in the unconditional formulation, there is only a single Z for all data instances. In conditional setting there is a $Z(z)$ for each data instance z.

Remarks. The temporal model of HSCRFs presented here is not a standard graphical model [14] since the connectivity (and therefore the clique structures) is not fixed. The potentials are defined on-the-fly depending on the context of assignments of ending indicators. Although the model topology is identical to that of shared structure HHMMs [15], the unrolled temporal representation is an undirected graph and the model distribution is formulated in a discriminative way. Furthermore, the state persistence potentials capture duration information that is not available in the dynamic DBN representation of the HHMMs in [17]. The HSCRF potentials may first appear to resemble the clique templates in the discriminative relational Markov networks [19]. They are, however, different because cliques in the HSCRFs are dynamic and context-dependent.

4. Asymmetric inside–outside algorithm

This section describes a core inference engine called Asymmetric Inside–Outside (AIO) algorithm. The AIO algorithm computes *building blocks* that are needed in inference and learning tasks, including the partition function, time-specific marginals and feature expectations.

4.1. Building blocks and conditional independence

4.1.1. Contextual Markov blankets

In this subsection we define elements that are building blocks for inference and learning. These building blocks are identified given the corresponding boundaries. Let us introduce two types of boundaries: the contextual *symmetric* and *asymmetric Markov blankets*.

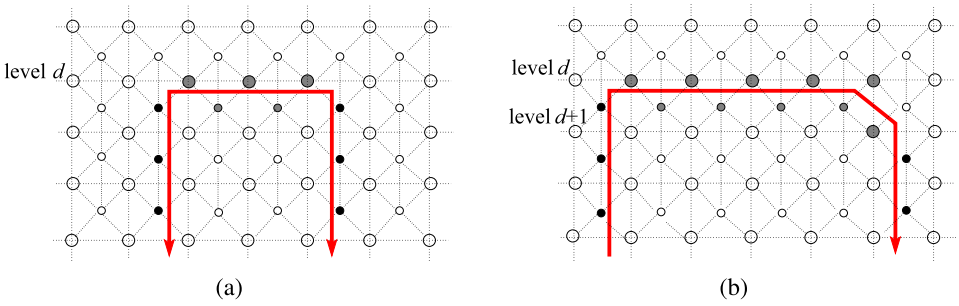

Fig. 7. (a) Symmetric Markov blanket, and (b) asymmetric Markov blanket.

Definition 1. A symmetric Markov blanket at level d for a state s starting at i and ending at j is the following set

$$\Pi_{i:j}^{d,s} = \left(x_{i:j}^{d} = s, e_{i-1}^{d:D} = 1, e_{j}^{d:D} = 1, e_{i:j-1}^{d} = 0\right) \tag{10}$$

Definition 2. Let $\Pi_{i:j}^{d,s}$ be a symmetric Markov blanket, we define $\zeta_{i:j}^{d,s}$ and $\underline{\zeta}_{i:j}^{d,s}$ as follows

$$\zeta_{i:j}^{d,s} = \left(x_{i:j}^{d+1:D}, e_{i:j-1}^{d+1:D}\right) \tag{11}$$

$$\underline{\zeta}_{i:j}^{d,s} = \zeta \setminus \left(\zeta_{i:j}^{d,s}, \Pi_{i:j}^{d,s}\right) \tag{12}$$

subject to $x_{i:j}^{d} = s$. Further, we define

$$\hat{\zeta}_{i:j}^{d,s} = \left(\zeta_{i:j}^{d,s}, \Pi_{i:j}^{d,s}\right) \quad \text{and} \quad \hat{\underline{\zeta}}_{i:j}^{d,s} = \left(\underline{\zeta}_{i:j}^{d,s}, \Pi_{i:j}^{d,s}\right)$$

Fig. 7a shows an example of a symmetric Markov blanket (represented by a double-arrowed line).

Definition 3. A asymmetric Markov blanket at level d for a parent state s starting at i and a child state u ending at j is the following set

$$\Gamma_{i:j}^{d,s}(u) = \left[x_{i:j}^{d} = s, x_{j}^{d+1} = u, e_{i-1}^{d:D} = 1, e_{j}^{d+1:D} = 1, e_{i:j-1}^{d} = 0\right] \tag{13}$$

Definition 4. Let $\Gamma_{i:j}^{d,s}(u)$ be an asymmetric Markov blanket, we define $\zeta_{i:j}^{d,s}(u)$ and $\underline{\zeta}_{i:j}^{d,s}(u)$ as follows

$$\zeta_{i:j}^{d,s}(u) = \left(x_{i:j-1}^{d+1:D}, x_{j}^{d+2:D}, e_{i:j-1}^{d+1:D}\right) \tag{14}$$

$$\underline{\zeta}_{i:j}^{d,s}(u) = \zeta \setminus \left(\zeta_{i:j}^{d,s}(u), \Gamma_{i:j}^{d,s}(u)\right) \tag{15}$$

subject to $x_{i:j}^{d} = s$ and $x_{j}^{d+1} = u$. Further, we define

$$\hat{\zeta}_{i:j}^{d,s}(u) = \left(\zeta_{i:j}^{d,s}(u), \Gamma_{i:j}^{d,s}(u)\right) \tag{16}$$

$$\hat{\underline{\zeta}}_{i:j}^{d,s}(u) = \left(\underline{\zeta}_{i:j}^{d,s}(u), \Gamma_{i:j}^{d,s}(u)\right) \tag{17}$$

Fig. 7b shows an example of asymmetric Markov blanket (represented by an arrowed line).

Remark. The concepts of contextual Markov blankets (or Markov blankets for short) are different from those in traditional Markov random fields and Bayesian networks because they are specific assignments of a subset of variables, rather than a collection of variables.

4.1.2. Conditional independence

Given these two definitions we have the following propositions of conditional independence.

Proposition 1. *$\zeta^{d,s}_{i:j}$ and $\underline{\zeta}^{d,s}_{i:j}$ are conditionally independent given $\Pi^{d,s}_{i:j}$*

$$\Pr\left(\zeta^{d,s}_{i:j}, \underline{\zeta}^{d,s}_{i:j} \mid \Pi^{d,s}_{i:j}\right) = \Pr\left(\zeta^{d,s}_{i:j} \mid \Pi^{d,s}_{i:j}\right) \Pr\left(\underline{\zeta}^{d,s}_{i:j} \mid \Pi^{d,s}_{i:j}\right) \tag{18}$$

This proposition gives rise to the following factorisation

$$\Pr(\zeta) = \Pr\left(\Pi^{d,s}_{i:j}\right) \Pr\left(\zeta^{d,s}_{i:j}, \underline{\zeta}^{d,s}_{i:j} \mid \Pi^{d,s}_{i:j}\right) = \Pr\left(\Pi^{d,s}_{i:j}\right) \Pr\left(\zeta^{d,s}_{i:j} \mid \Pi^{d,s}_{i:j}\right) \Pr\left(\underline{\zeta}^{d,s}_{i:j} \mid \Pi^{d,s}_{i:j}\right) \tag{19}$$

Proposition 2. *$\zeta^{d,s}_{i:j}(u)$ and $\underline{\zeta}^{d,s}_{i:j}(u)$ are conditionally independent given $\Gamma^{d,s}_{i:j}(u)$*

$$\Pr\left(\zeta^{d,s}_{i:j}(u), \underline{\zeta}^{d,s}_{i:j}(u) \mid \Gamma^{d,s}_{i:j}(u)\right) = \Pr\left(\zeta^{d,s}_{i:j}(u) \mid \Gamma^{d,s}_{i:j}(u)\right) \Pr\left(\underline{\zeta}^{d,s}_{i:j}(u) \mid \Gamma^{d,s}_{i:j}(u)\right) \tag{20}$$

The following factorisation is a consequence of Proposition 2

$$\begin{aligned}\Pr(\zeta) &= \Pr\left(\Gamma^{d,s}_{i:j}(u)\right) \Pr\left(\zeta^{d,s}_{i:j}(u), \underline{\zeta}^{d,s}_{i:j}(u) \mid \Gamma^{d,s}_{i:j}(u)\right) \\ &= \Pr\left(\Gamma^{d,s}_{i:j}(u)\right) \Pr\left(\zeta^{d,s}_{i:j}(u) \mid \Gamma^{d,s}_{i:j}(u)\right) \Pr\left(\underline{\zeta}^{d,s}_{i:j}(u) \mid \Gamma^{d,s}_{i:j}(u)\right)\end{aligned} \tag{21}$$

The proofs of Propositions 1 and 2 are given in Appendix A.1.

4.1.3. Symmetric inside/outside masses

From Eq. (12) we have $\zeta = \left(\zeta^{d,s}_{i:j}, \Pi^{d,s}_{i:j}, \underline{\zeta}^{d,s}_{i:j}\right)$. Since $\Pi^{d,s}_{i:j}$ separates $\zeta^{d,s}_{i:j}$ from $\underline{\zeta}^{d,s}_{i:j}$, we can group local potentials in Eq. (8) into three parts: $\Phi\left[\hat{\zeta}^{d,s}_{i:j}\right]$, $\Phi\left[\hat{\underline{\zeta}}^{d,s}_{i:j}\right]$, and $\Phi\left[\Pi^{d,s}_{i:j}\right]$. By 'grouping' we mean to multiply all the local potentials belonging to a certain part, in the same way that we group all the local potentials belonging to the model in Eq. (8). Note that although $\hat{\zeta}^{d,s}_{i:j}$ contains $\Pi^{d,s}_{i:j}$ we do not group $\Phi\left[\Pi^{d,s}_{i:j}\right]$ into $\Phi\left[\hat{\zeta}^{d,s}_{i:j}\right]$. The same holds for $\Phi\left[\hat{\underline{\zeta}}^{d,s}_{i:j}\right]$.

By definition of the state-persistence clique potential (Fig. 3), we have $\Phi\left[\Pi^{d,s}_{i:j}\right] = R^{d,s}_{i:j}$. Thus Eq. (8) can be replaced by

$$\Phi[\zeta] = \Phi\left[\hat{\zeta}^{d,s}_{i:j}\right] R^{d,s}_{i:j} \Phi\left[\hat{\underline{\zeta}}^{d,s}_{i:j}\right] \tag{22}$$

There are two special cases: (1) when $d = 1$, $\Phi\left[\hat{\underline{\zeta}}^{1,s}_{1:T}\right] = 1$ for $s \in S^1$, and (2) when $d = D$, $\Phi\left[\hat{\zeta}^{D,s}_{i:i}\right] = 1$ for $s \in S^D$ and $i \in [1, T]$. This factorisation plays an important role in efficient inference.

We know define a quantity called *symmetric inside mass* $\Delta^{d,s}_{i:j}$, and another called *symmetric outside mass* $\Lambda^{d,s}_{i:j}$.

Definition 5. Given a symmetric Markov blanket $\Pi^{d,s}_{i:j}$, the symmetric inside mass $\Delta^{d,s}_{i:j}$ and the symmetric outside mass $\Lambda^{d,s}_{i:j}$ are defined as

$$\Delta^{d,s}_{i:j} = \sum_{\zeta^{d,s}_{i:j}} \Phi\left[\hat{\zeta}^{d,s}_{i:j}\right] \tag{23}$$

$$\Lambda^{d,s}_{i:j} = \sum_{\underline{\zeta}^{d,s}_{i:j}} \Phi\left[\hat{\underline{\zeta}}^{d,s}_{i:j}\right] \tag{24}$$

As special cases we have $\Lambda^{1,s}_{1:T} = 1$ and $s \in S^1$, and $\Delta^{D,s}_{i:i} = 1$ for $i \in [1, T]$, $s \in S^D$. For later use let us introduce the 'full' symmetric inside mass $\hat{\Delta}^{d,s}_{i:j}$ and the 'full' symmetric outside mass $\hat{\Lambda}^{d,s}_{i:j}$ as

$$\hat{\Delta}^{d,s}_{i:j} = R^{d,s}_{i:j} \Delta^{d,s}_{i:j} \quad \text{and} \quad \hat{\Lambda}^{d,s}_{i:j} = R^{d,s}_{i:j} \Lambda^{d,s}_{i:j}$$

In the rest of the paper, when it is clear in the context, we will use *inside mass* as a shorthand for symmetric inside mass, *outside mass* for symmetric outside mass, *full-inside mass* for full-symmetric inside mass, and *full-outside mass* for full-symmetric outside mass.

Thus, from Eq. (22) the partition function can be computed from the full-inside mass at the top level ($d = 1$)

$$\begin{aligned}Z &= \sum_{\zeta} \Phi[\zeta] = \sum_{\zeta^{1,s}_{1:T}} \sum_{s \in S^1} \Phi\left[\hat{\zeta}^{1,s}_{1:T}\right] R^{1,s}_{1:T} = \sum_{s \in S^1} \Delta^{d,s}_{1:T} R^{d,s}_{1:T} \\ &= \sum_{s \in S^1} \hat{\Delta}^{1,s}_{1:T}\end{aligned} \tag{25}$$

Table 4
Computing the partition function from the full-inside mass and full-outside mass.

• $Z = \sum_{s \in S^1} \hat{\Delta}^{1,s}_{1:T}$
• $Z = \sum_{s \in S^D} \hat{\Lambda}^{D,s}_{i:i}$ for any $i \in [1, T]$
• $Z = \sum_{s \in S^d} \sum_{i \in [1,t]} \sum_{j \in [t,T]} \Delta^{d,s}_{i:j} \Lambda^{d,s}_{i:j} R^{d,s}_{i:j}$ for any $t \in [1, T]$ and $d \in [2, D-1]$

With the similar derivation the partition function can also be computed from the full-outside mass at the bottom level ($d = D$)

$$Z = \sum_{s \in S^D} \hat{\Lambda}^{D,s}_{i:i}, \text{ for any } i \in [1, T] \tag{26}$$

In fact, we will prove a more general way to compute Z in Appendix B

$$Z = \sum_{s \in S^d} \sum_{i \in [1,t]} \sum_{j \in [t,T]} \Delta^{d,s}_{i:j} \Lambda^{d,s}_{i:j} R^{d,s}_{i:j} \tag{27}$$

for any $t \in [1, T]$ and $d \in [2, D-1]$. These relations are summarised in Table 4.

Given the fact that $\zeta^{d,s}_{i:j}$ is separated from the rest of variables by the symmetric Markov blanket $\Pi^{d,s}_{i:j}$, we have Proposition 3.

Proposition 3. *The following relations hold*

$$\Pr\left(\zeta^{d,s}_{i:j} \mid \Pi^{d,s}_{i:j}\right) = \frac{1}{\Delta^{d,s}_{i:j}} \Phi\left[\hat{\zeta}^{d,s}_{i:j}\right] \tag{28}$$

$$\Pr\left(\underline{\zeta}^{d,s}_{i:j} \mid \Pi^{d,s}_{i:j}\right) = \frac{1}{\Lambda^{d,s}_{i:j}} \Phi\left[\hat{\underline{\zeta}}^{d,s}_{i:j}\right] \tag{29}$$

$$\Pr\left(\Pi^{d,s}_{i:j}\right) = \frac{1}{Z} \Delta^{d,s}_{i:j} R^{d,s}_{i:j} \Lambda^{d,s}_{i:j} \tag{30}$$

The proof of this proposition is given in Appendix A.2.

4.1.4. Asymmetric inside/outside masses

Recall that we have introduced the concept of asymmetric Markov blanket $\Gamma^{d,s}_{i:j}(u)$ which separates $\zeta^{d,s}_{i:j}(u)$ and $\underline{\zeta}^{d,s}_{i:j}(u)$. Let us group all the local contextual clique potentials associated with $\zeta^{d,s}_{i:j}(u)$ and $\Gamma^{d,s}_{i:j}(u)$ into a joint potential $\Phi\left[\hat{\zeta}^{d,s}_{i:j}(u)\right]$. Similarly, we group all local potentials associated with $\underline{\zeta}^{d,s}_{i:j}(u)$ and $\Gamma^{d,s}_{i:j}(u)$ into a joint potential $\Phi\left[\hat{\underline{\zeta}}^{d,s}_{i:j}(u)\right]$. Note that $\Phi\left[\hat{\underline{\zeta}}^{d,s}_{i:j}(u)\right]$ includes the state-persistence potential $R^{d,s}_{i:j}$.

Definition 6. Given the asymmetric Markov blanket $\Gamma^{d,s}_{i:j}(u)$, the asymmetric inside mass $\alpha^{d,s}_{i:j}(u)$ and the asymmetric outside mass $\lambda^{d,s}_{i:j}(u)$ are defined as follows

$$\alpha^{d,s}_{i:j}(u) = \sum_{\zeta^{d,s}_{i:j}(u)} \Phi\left[\hat{\zeta}^{d,s}_{i:j}(u)\right] \tag{31}$$

$$\lambda^{d,s}_{i:j}(u) = \sum_{\underline{\zeta}^{d,s}_{i:j}(u)} \Phi\left[\hat{\underline{\zeta}}^{d,s}_{i:j}(u)\right] \tag{32}$$

The relationship between the asymmetric outside mass and asymmetric inside mass is analogous to that between the outside and inside masses. However, there is a small difference, that is, the asymmetric outside mass 'owns' the segment $x^d_{i:j} = s$ and the associated state-persistence potential $R^{d,s}_{i:j}$, whilst the outside mass $\Lambda^d_{i:j}(s)$ does not.

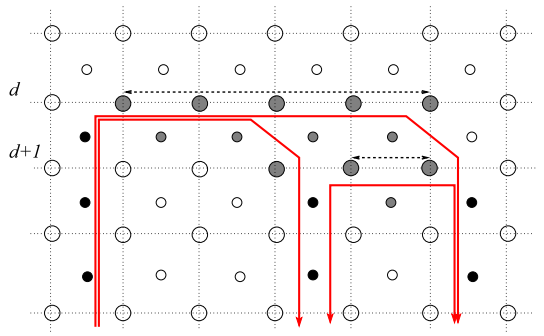

Fig. 8. Decomposition with respect to symmetric/asymmetric Markov blankets.

4.2. Computing inside masses

In this subsection we show how to recursively compute the pair: inside mass and asymmetric inside mass. The key idea here is to exploit the decomposition within the asymmetric Markov blanket. As shown in Fig. 8, an outer asymmetric Markov blanket can be decomposed into a sub-asymmetric Markov blanket and a symmetric blanket.

4.2.1. Computing asymmetric inside mass from inside mass

Assume that within the asymmetric Markov blanket $\Gamma^{d,s}_{i:j}(u)$, the child u starts somewhere at $t \in [i, j]$ and ends at j, i.e. $x^{d+1}_{t:j} = u$, $e^{d+1}_{t:j-1} = 0$ and $e^{d+1:D-1}_{t-1} = 1$. Let us consider two cases: $t > i$ and $t = i$.

Case 1. For $t > i$, denote by $v = x^{d+1}_{t-1}$. We have two smaller blankets within $\Gamma^{d,s}_{i:j}(u)$: the symmetric blanket $\Pi^{d+1,u}_{t:j}$ associated with the child $u = x^{d+1}_{t:j}$, and the asymmetric blanket $\Gamma^{d,s}_{i:t-1}(v)$ associated with the child v ending at $t-1$ under the parent s. Fig. 8 illustrates the blanket decomposition. The assignment $\zeta^{d,s}_{i:j}(u)$ can be decomposed as

$$\zeta^{d,s}_{i:j}(u) = \left(\zeta^{d,s}_{i:t-1}(v), \zeta^{d+1,u}_{t:j}, u = x^{d+1}_{t:j}, e^{d}_{t-1:j-1} = 0, e^{d+1:D}_{t-1} = 1\right) \tag{33}$$

Thus, the joint potential $\Phi\left[\hat{\zeta}^{d,s}_{i:j}(u)\right]$ can be factorised as follows

$$\Phi\left[\hat{\zeta}^{d,s}_{i:j}(u)\right] = \Phi\left[\hat{\zeta}^{d,s}_{i:t-1}(v)\right]\Phi\left[\hat{\zeta}^{d+1,u}_{t:j}\right] A^{d+1,s}_{v,u,t-1} R^{d+1,u}_{t:j} \tag{34}$$

The transition potential $A^{d+1,s}_{v,u,t-1}$ is enabled in the context $c = \left[e^{d}_{t-1} = 0, e^{d+1}_{t-1} = 1, x^{d}_{t} = s, x^{d+1}_{t-1} = v, x^{d+1}_{t} = u\right]$, and the state-persistence potential $R^{d+1,u}_{t:j}$ in the context $c = \left[e^{d+1}_{t:j-1} = 0, e^{d+1:D}_{t-1} = 1, e^{d+1:D}_{j} = 1, x^{d+1}_{t:j} = u\right]$.

Case 2. For $t = i$, the asymmetric blanket $\Gamma^{d,s}_{i:t-1}(v)$ does not exist since $i > t-1$. We have the following decompositions of assignment $\hat{\zeta}^{d,s}_{i:j}(u) = \left(\hat{\zeta}^{d+1,u}_{i:j}, e^{d}_{i-1} = 1, e^{d}_{i:j-1} = 0\right)$. In the context $c = \left[e^{d}_{i-1} = 1\right]$, the state-initialisation potential $\pi^{d,s}_{u,i}$ is activated. Thus we have

$$\Phi\left[\hat{\zeta}^{d,s}_{i:j}(u)\right] = \pi^{d,s}_{u,i}\Phi\left[\hat{\zeta}^{d+1,u}_{i:j}\right] R^{d+1,u}_{i:j} \tag{35}$$

Substituting Eqs. (34), (35) into Eq. (31), and together with the fact that t can take any value in the interval $[i, j]$, and v can take any value in S^{d+1}, we have the following relation

$$\begin{aligned}\alpha^{d,s}_{i:j}(u) &= \sum_{t\in[i+1,j]}\sum_{v\in S^{d+1}}\sum_{\zeta^{d,s}_{i:t-1}(v)}\sum_{\zeta^{d+1,u}_{t:j}} \Phi\left[\hat{\zeta}^{d,s}_{i:t-1}(v)\right]\Phi\left[\hat{\zeta}^{d+1,u}_{t:j}\right] A^{d+1,s}_{v,u,t-1} R^{d+1,u}_{t:j} + \sum_{\zeta^{d+1,u}_{i:j}} \pi^{d,s}_{u,i}\Phi\left[\hat{\zeta}^{d+1,u}_{i:j}\right] R^{d+1,u}_{i:j} \\ &= \sum_{t\in[i+1,j]}\sum_{v\in S^{d+1}} \alpha^{d,s}_{i:t-1}(v)\hat{\Delta}^{d+1,u}_{t:j} A^{d+1,s}_{v,u,t-1} + \hat{\Delta}^{d+1,u}_{i:j}\pi^{d,s}_{u,i}\end{aligned} \tag{36}$$

As we can see, the asymmetric inside mass α plays the role of a *forward message* starting from the starting time i to the ending time j. There is a recursion where the asymmetric inside mass ending at time j is computed from all the asymmetric inside masses ending at time $t-1$, for $t \in [i+1, j]$.

There are special cases for the asymmetric inside mass: (1) when $i = j$, we only have

$$\alpha^{d,s}_{i:i}(u) = \hat{\Delta}^{d+1,s}_{i:i}\pi^{d,s}_{u,i} \tag{37}$$

and (2) when $d = D-1$, the sum over the index t as in Eq. (36) is not allowed since at level D the inside mass only spans a single index. We have the following instead

$$\begin{aligned}\alpha_{i:j}^{D-1,s}(u) &= \sum_{v\in S^{d+1}} \alpha_{i:j-1}^{D-1,s}(v)\hat{\Delta}_{j:j}^{D,u} A_{v,u,j-1}^{D,s} \\ &= \sum_{v\in S^{d+1}} \alpha_{i:j-1}^{D-1,s}(v) R_{j:j}^{D,u} A_{v,u,j-1}^{D,s} \end{aligned} \tag{38}$$

4.2.2. Computing inside mass from asymmetric inside mass

Notice the relationship between the asymmetric Markov blanket $\Gamma_{i:j}^{d,s}(u)$ and the symmetric blanket $\Pi_{i:j}^{d,s}$, where $d < D$. When $e_j^d = 1$, i.e. the parent s ends at j, and $\Gamma_{i:j}^{d,s}(u)$ will become $\Pi_{i:j}^{d,s}$ with $u = x_j^{d+1}$. Then we have decompositions $\zeta_{i:j}^{d,s} = \left(\zeta_{i:j}^{d,s}(u), u = x_j^{d+1}\right)$ and $\hat{\zeta}_{i:j}^{d,s} = \left(\hat{\zeta}_{i:j}^{d,s}(u), e_j^d = 1, u = x_j^{d+1}\right)$. These lead to the factorisation

$$\Phi\left[\hat{\zeta}_{i:j}^{d,s}\right] = \Phi\left[\hat{\zeta}_{i:j}^{d,s}(u)\right] E_{u,j}^{d,s} \tag{39}$$

where the state-ending potential $E_{u,j}^{d,s}$ is activated in the context $c = \left[e_j^d = 1\right]$. Thus, the inside mass in Eq. (23) can be rewritten as

$$\begin{aligned}\Delta_{i:j}^{d,s} &= \sum_{u\in S^{d+1}} \sum_{\zeta_{i:j}^{d,s}(u)} \Phi[\hat{\zeta}_{i:j}^{d,s}(u)] E_{u,j}^{d,s} \\ &= \sum_{u\in S^{d+1}} E_{u,j}^{d,s} \sum_{\zeta_{i:j}^{d,s}(u)} \Phi[\hat{\zeta}_{i:j}^{d,s}(u)] \\ &= \sum_{u\in S^{d+1}} E_{u,j}^{d,s} \alpha_{i:j}^{d,s}(u) \end{aligned} \tag{40}$$

This equation holds for $d < D$. When $d = D$, we set $\Delta_{i:i}^{D,s} = 1$ for all $s \in S^D$ and $i \in [1, T]$, and when $d = 1$, we must ensure that $i = 1$ and $j = T$.

Remark. Eqs. (36), (37), (38) and (40) specify a *left-right* and *bottom-up* algorithm to compute both the inside and asymmetric inside masses. Initially, at the bottom level $\Delta_{i:i}^{D,s} = 1$ for $i \in [1, T]$ and $s \in S^D$. Pseudo-code of the dynamic programming algorithm to compute all the inside and asymmetric inside masses and the partition function is given in Algorithm 1.

Algorithm 1 Computing the set of inside/asymmetric inside masses and the partition function.

Input: D, T, all the potential function values.

Output: partition function Z; $\Delta_{1:T}^{1,s}$, for $s \in S^1$;$\Delta_{i:i}^{D,s}$ for $s \in S^D$ and $i \in [1, T]$;
$\Delta_{i:j}^{d,s}$, for $d \in [2, D-1]$, $s \in S^d$ and $1 \le i \le j \le T$;
$\alpha_{i:j}^{d,s}(u)$ for $d \in [1, D-1]$, $u \in S^{d+1}$ and $1 \le i \le j \le T$

Initialise: $\Delta_{i:i}^{D,s} = 1$ for all $i \in [1, T]$ and $s \in S^D$
/* *At the level d=D-1* */
For $i = 1, 2, ..., T$; $j = i, i+1, ..., T$
  Compute $\alpha_{i:j}^{D-1,s}(u)$ using Eq. (38)
  Compute $\Delta_{i:j}^{D-1,s}$ using Eq. (40)
EndFor
/* *The main recursion loops: bottom-up and forward* */
For $d = D-2, D-3, ..., 1$
  For $i = 1, 2, ..., T$; $j = i, i+1, ..., T$
    Compute $\alpha_{i:i}^{d,s}(u)$ using Eq. (37) **If** $j = i$
    Compute $\alpha_{i:j}^{d,s}(u)$ using Eq. (36) **If** $j > i$
    Compute $\Delta_{i:j}^{d,s}$ using Eq. (40) **If** $d > 1$
  EndFor
EndFor
Compute Z using Eq. (25).

4.3. Computing outside masses

In this subsection we show how to recursively compute the symmetric outside mass and the asymmetric outside mass. We use the same blanket decomposition as in Section 4.2. However, this time the view is reversed as we are interested in quantities outside the blankets. For example, outside the inner symmetric Markov blanket in Fig. 8, there exists an outer asymmetric blanket and another sub-asymmetric blanket on the left.

4.3.1. Computing asymmetric outside mass from outside mass

Let us examine the variables $\underline{\zeta}^{d,s}_{i:j}(u)$ associated with the asymmetric Markov blanket $\Gamma^{d,s}_{i:j}(u)$, for $d \in [1, D-1]$ and $1 \le i \le j \le T$ (see Definition 4). For $j < T$, assume that there exists an outer asymmetric Markov blanket $\Gamma^{d,s}_{i:t}(v)$ for some $v \in S^{d+1}$ and $t \in [j+1, T]$, and a symmetric Markov blanket $\Pi^{d+1,v}_{j+1:t}$ right next to $\Gamma^{d,s}_{i:j}(u)$. Given these blankets we have the decomposition $\hat{\underline{\zeta}}^{d,s}_{i:j}(u) = \left(\hat{\underline{\zeta}}^{d,s}_{i:t}(v), \hat{\zeta}^{d+1,v}_{j+1:t}, x^{d+1}_j = u\right)$, which leads to the following factorisation

$$\Phi\left[\hat{\underline{\zeta}}^{d,s}_{i:j}(u)\right] = \Phi\left[\hat{\underline{\zeta}}^{d,s}_{i:t}(v)\right] \Phi\left[\hat{\zeta}^{d+1,v}_{j+1:t}\right] R^{d+1,v}_{j+1:t} A^{d+1,s}_{u,v,j} \tag{41}$$

The state transition potential $A^{d+1,s}_{u,v,j}$ is enabled in the context $c = \left[e^d_j = 0, e^{d+1}_j = 1\right]$, and the state persistence potential $R^{d+1,v}_{j+1:t}$ in the context $c = \left[e^{d+1}_j = 1, e^{d+1}_{j+1:t-1} = 0, e^{d+1}_t = 1\right]$.

In addition, there exists a special case where the state s ends at j. We have the decomposition $\hat{\underline{\zeta}}^{d,s}_{i:j}(u) = \left(\hat{\underline{\zeta}}^{d,s}_{i:j}, u = x^{d+1}_j\right)$ and the following factorisation

$$\Phi\left[\hat{\underline{\zeta}}^{d,s}_{i:j}(u)\right] = \Phi\left[\hat{\underline{\zeta}}^{d,s}_{i:j}\right] R^{d,s}_{i:j} E^{d,s}_{u,j} \tag{42}$$

The ending potential $E^{d,s}_{u,j}$ appears here because of the context $c = \left[e^d_j = 1\right]$, i.e. s ends at j.

Now we relax the assumption of t, v and allow them to receive all possible values, i.e. $t \in [j, T]$ and $v \in S^{d+1}$. Thus we can replace Eq. (32) by

$$\begin{aligned}\lambda^{d,s}_{i:j}(u) &= \sum_{v \in S^{d+1}} \sum_{t \in [j+1,T]} \sum_{\underline{\zeta}^{d,s}_{i:t}(v)} \sum_{\zeta^{d+1,v}_{j+1:t}} \Phi\left[\hat{\underline{\zeta}}^{d,s}_{i:t}(v)\right] \Phi\left[\hat{\zeta}^{d+1,v}_{j+1:t}\right] R^{d+1,v}_{j+1:t} A^{d+1,s}_{u,v,j} + \sum_{\underline{\zeta}^{d,s}_{i:j}(u)} \Phi\left[\hat{\underline{\zeta}}^{d,s}_{i:j}\right] R^{d,s}_{i:j} E^{d,s}_{u,j} \\ &= \sum_{v \in S^{d+1}} \sum_{t \in [j+1,T]} \lambda^{d,s}_{i:t}(v) \hat{\Delta}^{d+1,v}_{j+1:t} A^{d+1,s}_{u,v,j} + \hat{\Lambda}^{d,s}_{i:j} E^{d,s}_{u,j}\end{aligned} \tag{43}$$

for $d \in [2, D-2]$, and $1 \le i \le j \le T$. Thus, the $\lambda^{d,s}_{i:j}(u)$ can be thought as a message passed *backward* from $j = T$ to $j = i$. Here, the asymmetric outside mass ending at j is computed by using all the asymmetric outside masses ending at t for $t \in [j+1, T]$.

There are two special cases. At the top level, i.e. $d = 1$, then $\lambda^{d,s}_{i:j}(u)$ is only defined at $i = 1$, and the second term of the RHS of Eq. (43) is included only if $i = 1, j = T$. At the second lowest level, i.e. $d = D-1$, we cannot sum over t as in Eq. (43) since $\hat{\Delta}^{D,v}_{j+1:t}$ is only defined for $t = j+1$. We have the following relation instead

$$\lambda^{D-1,s}_{i:j}(u) = \sum_{v \in S^D} \lambda^{D-1,s}_{i:j+1}(v) \hat{\Delta}^{D,v}_{j+1:j+1} A^{D,s}_{u,v,j} + \hat{\Lambda}^{D-1,s}_{i:j} E^{D-1,s}_{u,j} \tag{44}$$

4.3.2. Computing outside mass from asymmetric outside mass

Given a symmetric Markov blanket $\Pi^{d+1,u}_{i:j}$ for $d \in [1, D-1]$, assume that there exists an asymmetric Markov blanket $\Gamma^{d,s}_{t:j}(u)$ at the parent level d, where $t \in [1, i]$. Clearly, for $t \in [1, i-1]$ there exists some sub-asymmetric Markov blanket $\Gamma^{d,s}_{t:i-1}(v)$. See Fig. 8 for an illustration.

Let us consider two cases: $t < i$ and $t = i$.

Case 1. For $t < i$, this enables the decomposition $\hat{\underline{\zeta}}^{d+1,u}_{i:j} = \left(\hat{\underline{\zeta}}^{d,s}_{t:j}(u), \hat{\zeta}^{d,s}_{t:i-1}(v), u = x^{d+1}_{i:j}\right)$, which leads to the following factorisation

$$\Phi\left[\hat{\underline{\zeta}}^{d+1,u}_{i:j}\right] = \Phi\left[\hat{\underline{\zeta}}^{d,s}_{t:j}(u)\right] \Phi\left[\hat{\zeta}^{d,s}_{t:i-1}(v)\right] A^{d,s}_{v,u,i-1} \tag{45}$$

The state transition potential $A^{d,s}_{v,u,i-1}$ is activated in the context $c = \left[e^d_{i-1} = 0, e^{d+1}_{i-1} = 1\right]$.

Table 5
Summary of basic building blocks computed in Section 4.2 and 4.3.

- $\Delta^{1,s}_{1:T}, \Lambda^{1,s}_{1:T}$ for $s \in S^1$
- $\Delta^{d,s}_{i:j}, \Lambda^{d,s}_{i:j}$ for $d \in [2, D-1], s \in S^d, 1 \le i \le j \le T$
- $\Delta^{D,s}_{i:i}, \Lambda^{D,s}_{i:i}$ for $i \in [1, T], s \in S^D$
- $\alpha^{d,s}_{1:j}(u), \lambda^{d,s}_{1:j}(u)$ for $d = 1, s \in S^1, u \in S^2, j \in [1, T]$
- $\alpha^{d,s}_{i:j}(u), \lambda^{d,s}_{i:j}(u)$ for $d \in [2, D-1], s \in S^d, u \in S^{d+1}, 1 \le i \le j \le T$

Case 2. For $t = i$, the decomposition reduces to $\hat{\underline{\zeta}}^{d+1,u}_{i:j} = \left(\hat{\underline{\zeta}}^{d,s}_{i:j}(u), u = x^{d+1}_{i:j}\right)$, which leads to the following factorisation

$$\Phi\left[\hat{\underline{\zeta}}^{d+1,u}_{i:j}\right] = \Phi\left[\hat{\underline{\zeta}}^{d,s}_{i:j}(u)\right]\pi^{d,s}_{u,i} \tag{46}$$

The state-initialisation potential $\pi^{d,s}_{u,i}$ plays the role in the context $c = \left[e^d_{i-1} = 1\right]$

However, these decompositions and factorisations only hold given the assumption of specific values of $s \in S^d$, $v \in S^{d+1}$, and $t \in [1, i]$. Without further information we have to take all possibilities into account. Substituting these relations into Eq. (24), we have

$$\begin{aligned}\Lambda^{d+1,u}_{i:j} &= \sum_{s\in S^d}\sum_{v\in S^{d+1}}\sum_{t\in[1,i-1]}\sum_{\underline{\zeta}^{d,s}_{t:j}(u)}\sum_{\underline{\zeta}^{d,s}_{t:i-1}(v)}\Phi\left[\hat{\underline{\zeta}}^{d,s}_{t:j}(u)\right]\Phi\left[\hat{\underline{\zeta}}^{d,s}_{t:i-1}(v)\right]A^{d+1,s}_{v,u,i-1} + \sum_{s\in S^d}\sum_{\underline{\zeta}^{d,s}_{i:j}(u)}\Phi\left[\hat{\underline{\zeta}}^{d,s}_{i:j}(u)\right]\pi^{d,s}_{u,i} \\ &= \sum_{s\in S^d}\sum_{t\in[1,i-1]}\lambda^{d,s}_{t:j}(u)\sum_{v\in S^{d+1}}\alpha^{d,s}_{t:i-1}(v)A^{d+1,s}_{v,u,i-1} + \sum_{s\in S^d}\lambda^{d,s}_{i:j}(u)\pi^{d,s}_{u,i}\end{aligned} \tag{47}$$

for $d \in [2, D-2]$.

There are three special cases. The first is the base case where $d = 0$ and $\Lambda^{1,s}_{1:T} = 1$ for all $s \in S^1$. In the second case, for $d = 1$, we must fix the index $t = 1$ since the asymmetric inside mass $\alpha^{d,s}_{t:i-1}$ is only defined at $t = 1$. Also the second term in the RHS is included only if $i = 1$ for the asymmetric outside mass $\lambda^{d,s}_{i:j}(u)$ to make sense. In the second case, for $d + 1 = D$, we only have $i = j$.

Remark. Eqs. (43), (44) and (47) show a recursive *top-down* and *outside-in* approach to compute the symmetric/asymmetric outside masses. We start from the top with $d = 1$ and $\Lambda^{1,s}_{1:T} = 1$ for all $s \in S^1$ and proceed downward until $d = D$. The pseudo-code is given in Algorithm 2. Table 5 summarises the quantities computed in Section 4.2 and 4.3.

Algorithm 2 Computing the set of outside/asymmetric outside masses.

Input: D, T, all the potential function values, all inside/asymmetric inside masses.
Output: all outside/asymmetric outside masses

Initialise: $\Lambda^{1,s}_{1:T} = 1$; $\lambda^{1,s}_{1:T}(u) = E^{1,s}_{u,T}$ for $s \in S^1, u \in S^2$
/ the main recursive loops: top-down and inside-out */*
For $d = 1, 2, ..., D-1$
  For $i = 1, 2, ..., T$; $j = T, T-1, ..., i$
    Compute the asymmetric outside mass $\lambda^{d,s}_{i:j}(u)$ using Eqs. (43), (44)
    Compute the outside mass $\Lambda^{d,s}_{i:j}$ using Eq. (47)
  EndFor
EndFor

Algorithm 3 summarises the AIO algorithm for computing all building blocks and the partition function.

Algorithm 3 The AIO algorithm.

Input: D, T, all the potential function values
Output: all building blocks and partition function

Compute all inside/asymmetric inside masses using the algorithm in Algorithm 1
Compute all outside/asymmetric outside masses using the algorithm in Algorithm 2

Table 6
Notations used in this section.

Notation	Description
$\Delta_{i:j}^{\max,d,s}$	The optimal potential function of the subset of variables $\zeta_{i:j}^{d,s}$
$\hat{\Delta}_{i:j}^{\max,d,s}$	The 'full' version of $\Delta_{i:j}^{\max,d,s}$
$\alpha_{i:j}^{\max,d,s}(u)$	The optimal potential function of the subset of variables $\zeta_{i:j}^{d,s}(u)$
$\Delta_{i:j}^{\arg,d,s}$	The optimal child u_j^{d+1} of s
$\alpha_{i:j}^{\arg,d,s}(u)$	The optimal child v_{t-1}^{d+1} that transits to $u_{t:j}^{d+1}$ and the time index t.
$\mathscr{I}^d$	The set of optimal 'segments' at each level d.

5. The generalised Viterbi algorithm

The MAP assignment is

$$\zeta^{MAP} = \arg\max_{\zeta} \Pr(\zeta \mid z) = \arg\max_{\zeta} \Phi[\zeta, z]$$

The process of computing the MAP assignment is similar to that of computing the partition function. This similarity comes from the relation between the sum-product and max-product algorithm (a generalisation of the Viterbi algorithm) of [20], and from the fact that inside/asymmetric inside procedures described in Section 4.2 are essentially a sum-product. We just need to convert all the summations into corresponding maximisations. The algorithm is a two-step procedure:

- In the first step the maximum joint potential is computed and local maximum states and ending indicators are saved along the way. These states and ending indicators are maintained in a *bookkeeper*.
- In the second step we decode the best assignment by *backtracking* through saved local maximum states.

We make use of the contextual decompositions and factorisations from Section 4.2.

Notations

This section, with some abuse of notations, uses some slight modifications to the notations used in the rest of the paper. See Table 6 for reference.

We now describe the first step.

5.1. Computing the maximum joint potential, maximal states and time indices

For clarity, let us drop the notation z in $\Phi[\zeta, z]$. As $\Phi[\zeta] = \Phi\left[\hat{\zeta}_{1:T}^{1,s}\right] R_{1:T}^{1,s}$ for $s \in S^1$ we have

$$\max_{\zeta} \Phi[\zeta] = \max_{s \in S^1} R_{1:T}^{1,s} \max_{\zeta_{1:T}^{1,s}} \Phi\left[\hat{\zeta}_{1:T}^{1,s}\right] \tag{48}$$

Now, for a sub-assignment $\zeta_{i:j}^{d,s}$ for $1 \in [1, D-1]$, Eq. (39) leads to

$$\max_{\zeta_{i:j}^{d,s}} \Phi\left[\hat{\zeta}_{i:j}^{d,s}\right] = \max_{u \in S^{d+1}} E_{u,j}^{d,s} \max_{\zeta_{i:j}^{d,s}(u)} \Phi\left[\hat{\zeta}_{i:j}^{d,s}(u)\right] \tag{49}$$

With some slight abuse of notation we introduce $\Delta_{i:j}^{\max,d,s}$ as the optimal potential function of the subset of variables $\zeta_{i:j}^{d,s}$, and $\alpha_{i:j}^{\max,d,s}(u)$ as the optimal potential function of the subset of variables $\zeta_{i:j}^{d,s}(u)$.

Definition 7. We define $\Delta_{i:j}^{\max,d,s}$ and $\alpha_{i:j}^{\max,d,s}(u)$ as follows

$$\Delta_{i:j}^{\max,d,s} = \max_{\zeta_{i:j}^{d,s}} \Phi\left[\hat{\zeta}_{i:j}^{d,s}\right] \tag{50}$$

$$\hat{\Delta}_{i:j}^{\max,d,s} = \Delta_{i:j}^{\max,d,s} R_{i:j}^{d,s} \tag{51}$$

$$\alpha_{i:j}^{\max,d,s}(u) = \max_{\zeta_{i:j}^{d,s}(u)} \Phi\left[\hat{\zeta}_{i:j}^{d,s}(u)\right] \tag{52}$$

Eqs. (48) and (49) can be rewritten more compactly as

$$\Phi\left[\zeta^{MAP}\right] = \max_{s\in S^1} \hat{\Delta}_{1:T}^{\max,1,s} \tag{53}$$

$$\Delta_{i:j}^{\max,d,s} = \max_{u\in S^{d+1}} E_{u,j}^{d,s} \alpha_{i:j}^{\max,d,s}(u) \tag{54}$$

for $d \in [1, D-1]$. When $d = D$, we simply set $\Delta_{i:i}^{\max,D,s} = 1$ for all $s \in S^D$ and $i \in [1, T]$.

From the factorisation in Eqs. (34), (35), we have

$$\max_{\zeta_{i:j}^{d,s}(u)} \Phi\left[\hat{\zeta}_{i:j}^{d,s}(u)\right] = \max\left\{\left(\max_{v\in S^{d+1}} \max_{t\in[i+1,j]} R_{t:j}^{d+1,u} A_{v,u,t-1}^{d+1,s} \max_{\zeta_{i:t-1}^{d,s}(v)} \Phi\left[\hat{\zeta}_{i:t-1}^{d,s}(v)\right] \times \right.\right.$$
$$\left.\left. \times \max_{\zeta_{t:j}^{d+1,u}} \Phi\left[\hat{\zeta}_{t:j}^{d+1,u}\right]\right); \left(R_{i:j}^{d+1,u} \max_{\zeta_{i:j}^{d+1,u}} \pi_{u,i}^{d,s} \Phi\left[\hat{\zeta}_{i:j}^{d+1,u}\right]\right)\right\} \tag{55}$$

and

$$\alpha_{i:j}^{\max,d,s}(u) = \max\left\{\left(\max_{v\in S^{d+1}} \max_{t\in[i+1,j]} \alpha_{i:t-1}^{\max,d,s}(v) \hat{\Delta}_{t:j}^{\max,d+1,u} A_{v,u,t-1}^{d,s}\right); \left(\hat{\Delta}_{i:j}^{\max,d+1,u} \pi_{u,i}^{d+1,s}\right)\right\} \tag{56}$$

for $d \in [1, D-2]$ and $i < j$. For $d = D-1$, we cannot scan the index t in the interval $[i+1, j]$ because the maximum inside $\Delta_{t:j}^{\max,D,u}$ is only defined at $t = j$. We have the following instead

$$\alpha_{i:j}^{\max,D-1,s}(u) = \max_{v\in S^D} \alpha_{i:j-1}^{\max,D-1,s}(v) \hat{\Delta}_{j:j}^{\max,D,u} A_{v,u,j-1}^{D,s} \tag{57}$$

There is a base case for $i = j$, where the context $c = \left[e_{i-1}^d = 1\right]$ is active, then

$$\alpha_{i:i}^{\max,d,s}(u) = \hat{\Delta}_{i:i}^{\max,d+1,u} \pi_{u,i}^{d,s} \tag{58}$$

Of course, what we are really interested in is not the maximum joint potentials but the optimal states and time indices (or ending indicators). We need some bookkeepers to hold these quantities along the way. With some abuse of notation let us introduce the symmetric inside bookkeeper $\Delta_{i:j}^{\arg,d,s}$ associated with Eq. (54), and the asymmetric inside bookkeeper $\alpha_{i:j}^{\arg,d,s}(u)$ associated with Eqs. (56), (57) and (58).

Definition 8. We define the symmetric inside bookkeeper $\Delta_{i:j}^{\arg,d,s}$ as follows

$$\Delta_{i:j}^{\arg,d,s} = u^* = \arg\max_{u\in S^{d+1}} E_{u,j}^{d,s} \alpha_{i:j}^{\max,d,s}(u) \tag{59}$$

Similarly, we define the asymmetric inside bookkeeper $\alpha_{i:j}^{\arg,d,s}(u)$ associated with Eq. (56) for $d \in [1, D-2]$ as

$$\alpha_{i:j}^{\arg,d,s}(u) = (v,t)^* = \arg\max_{t\in[i+1,j], v\in S^{d+1}} \alpha_{i:t-1}^{\max,d,s}(v) \hat{\Delta}_{t:j}^{\max,d+1,u} A_{v,u,t-1}^{d,s} \tag{60}$$

if $\max_{v\in S^{d+1}, t\in[i+1,j]} \alpha_{i:t-1}^{\max,d,s}(v) \hat{\Delta}_{t:j}^{\max,d+1,u} A_{v,u,t-1}^{d,s} > \hat{\Delta}_{i:j}^{\max,d+1,u} \pi_{u,i}^{d+1,s}$ and $i < j$; and

$$\alpha_{i:j}^{\arg,d,s}(u) = \text{ undefined} \tag{61}$$

otherwise. For $d = D-1$, the $\alpha_{i:j}^{\arg,d,s}(u)$ is associated with Eq. (57)

$$\alpha_{i:j}^{\arg,D-1,s}(u) = \arg\max_{v\in S^D} \alpha_{i:j-1}^{\max,d,s}(v) \hat{\Delta}_{j:j}^{\max,D,u} A_{v,u,j-1}^{d,s} \tag{62}$$

The Eqs. (53), (54), (56), (57) and (58) provide a recursive procedure to compute maximum joint potential in a bottom-up and left-right manner. Initially we just set $\Delta_{i:i}^{\max,D,s} = 1$ for all $s \in S^D$ and $i \in [1, T]$. The procedure is summarised in Algorithm 4.

Algorithm 4 Computing the bookkeepers.

Input: D, T, all the potential function values.

Output: the bookkeepers; $\Delta_{1:T}^{\arg,1,s}$, for $s \in S^1$ and $1 \leq i \leq j \leq T$;
$\Delta_{i:j}^{\arg,d,s}$, for $d \in [2, D-1]$, $s \in S^d$; $\Delta_{i:i}^{\arg,D,s}$ for $s \in S^D$ and $i \in [1, T]$;
$\alpha_{i:j}^{\arg,d,s}(u)$ for $d \in [1, D-1]$, $u \in S^{d+1}$ and $1 \leq i \leq j \leq T$

Initialise: $\Delta_{i:i}^{\max,D,s} = 1$ for all $i \in [1, T]$ and $s \in S^D$

/ At the level d=D-1 */*

For $i = 1, 2, ..., T$; $j = i, i+1, ..., T$

Compute $\alpha_{i:j}^{\max,D-1,s}(u)$ using Eq. (57) and $\alpha_{i:j}^{\arg,D-1,s}(u)$ using Eq. (62)

Compute $\Delta_{i:j}^{\max,D-1,s}$ using Eq. (54) and $\Delta_{i:j}^{\arg,D-1,s}$ using Eq. (59)

EndFor

/ The main recursion loops: bottom-up and forward */*

For $d = D-2, D-3, ..., 1$

For $i = 1, 2, ..., T$; $j = i, i+1, ..., T$

If $j = i$

Compute $\alpha_{i:i}^{\max,d,s}(u)$ using Eq. (58)

Else

Compute $\alpha_{i:j}^{\max,d,s}(u)$ using Eq. (56) and $\alpha_{i:i}^{\arg,d,s}(u)$ using Eq. (60)

EndIf

If $d > 1$

Compute $\Delta_{i:j}^{\max,d,s}$ using Eq. (54) and $\Delta_{i:j}^{\arg,d,s}$ using Eq. (59)

EndIf

EndFor

EndFor

Compute $\Delta_{1:T}^{\max,1,s}$ using Eq. (54) and $\Delta_{1:T}^{\arg,1,s}$ using Eq. (59)

5.2. Decoding the MAP assignment

The proceeding of the backtracking process is opposite to that of the max-product. Specifically, we start from the root and proceed in a *top-down* and *right-left* manner. The goal is to identify the right-most segment at each level. Formally, a segment is a triple (s, i, j) where s is the segment label, and i and j are start and end time indices, respectively. From the maximum inside $\Delta_{i:j}^{\max,d,s}$ at level d, we identify the best child u and its ending time j from Eq. (54). This gives rise to the maximum asymmetric inside $\alpha_{i:j}^{\max,d,s}(u)$. Then we seek for the best child v that transits to u under the same parent s using Eq. (56). Since the starting time t for u has been identified the ending time for v is $t-1$. We now have a right-most segment (u, t, j) at level $d+1$. The procedure is repeated until we reach the starting time i of the parent s. The backtracking algorithm is summarised in Algorithm 5.

Finally, the generalised Viterbi algorithm is given in Algorithm 6.

Working in log-space to avoid numerical overflow

With long sequence and complex topology we may run into the problem of numerical overflow, i.e. when the numerical value of the maximum joint potential is beyond the number representation of the machine. To avoid this, we can work in the log-space instead, using the monotonic property of the log function. The equations in the log-space are summarised in Table 7.

6. Parameter estimation

In this section, we tackle the problem of parameter estimation by maximising the (conditional) data likelihood. Typically we need some parametric form to be defined for a particular problem and we need some numerical method to do the optimisation task.

Here we employ the log-linear parameterisation, which is commonly used in the CRF setting. Recall from Section 2.1 that estimating parameters of the log-linear models using gradient-based methods requires the computation of feature expectation, or expected sufficient statistics (ESS). For our HSCRFs we need to compute four types of ESS corresponding to the state-persistence, state-transition, state-initialisation and state-ending.

Algorithm 5 Backtracking for optimal assignment (nested Markov blankets).

Input: D, T, all the filled bookkeepers.
Output: the optimal assignment ζ^{MAP}

$s^* = \arg\max_{s \in S^1} \hat{\Delta}^{\max,1,s}_{1:T}$
Initialise triple buckets $\mathscr{J}^1 = \{(s^*, 1, T)\}$ and $\mathscr{J}^d = \{\}$ for $d \in [2, D]$
For $d = 1, 2, ..., D-1$
  For each triple (s^*, i, j) in $\mathscr{J}^d$
    Let $u^* = \Delta^{\arg,d,s^*}_{i:j}$
    For $i \le j$
      If $\alpha^{\arg,d,s^*}_{i:j}(u^*)$ is defined **Then**
        $(t^*, v^*) = \alpha^{\arg,d,s^*}_{i:j}(u^*)$
        Add the triple (v^*, t^*, j) to $\mathscr{J}^{d+1}$ and Set $j = t^* - 1$ and $u^* = v^*$
      Else
        Add the triple (u^*, i, j) to $\mathscr{J}^{d+1}$ and Break this loop
      EndIf
    EndFor
  EndFor
EndFor
For each stored triple (s^*, i, j) in the bucket $\mathscr{J}^d$, for $d \in [1, D]$,
create a corresponding set of variables $(x^d_{i:j} = s^*, e^d_{i-1} = 1, e^d_j = 1, e^d_{i:j-1} = 0)$.
The joining of these sets is the optimal assignment ζ^{MAP}

Algorithm 6 The generalised Viterbi algorithm.

Input: D, T, all the potential function values.
Output: the optimal assignment ζ^{MAP}

Run the bottom-up discrete optimisation procedure described in Algorithm 4.
Run the top-down backtracking procedure described in Algorithm 5.

Table 7
MAP equations in the log-space.

Log-space equations	Equations
$\log \Delta^{\max,d,s}_{i:j} = \max_{u \in S^{d+1}} \{\log E^{d,s}_{u,j} + \log \alpha^{\max,d,s}_{i:j}(u)\}$	Eq. (54)
$\log \alpha^{\max,d,s}_{i:j}(u) = \max \Big\{ \max_{t \in [i+1,j]} \max_{v \in S^{d+1}} \{\log \alpha^{\max,d,s}_{i:t-1}(v) + \log \hat{\Delta}^{\max,d+1,u}_{t:j} + \log A^{d,s}_{v,u,t-1}\};\ \log \hat{\Delta}^{\max,d+1,u}_{i:j} + \log \pi^{d+1,s}_{u,i} \Big\}$	Eq. (56)
$\log \alpha^{\max,D-1,s}_{i:j}(u) = \max_{v \in S^D} \{\log \alpha^{\max,D-1,s}_{i:j-1}(v) + \log \hat{\Delta}^{\max,D,u}_{j:j} + \log A^{D,s}_{v,u,j-1}\}$	Eq. (57)
$\log \alpha^{\max,d,s}_{i:i}(u) = \log \hat{\Delta}^{\max,d+1,u}_{i:i} + \log \pi^{d,s}_{u,i}$	Eq. (58)

6.1. Log-linear parameterisation

In our HSCRF setting there is a feature vector $\mathbf{f}^d_\sigma(\sigma, z)$ associated with each type of contextual clique σ, in that $\phi(\sigma^d, z) = \exp\left[\mathbf{w}^\top_{\sigma^d} \mathbf{f}^d_\sigma(\sigma, z)\right]$. Thus, the features are active only in the context in which the corresponding contextual cliques appear.

For the state-persistence contextual clique, the features incorporate *state-duration*, start time i and end time j of the state. Other feature types incorporate the time index in which the features are triggered. Specifically,

$$R^{d,s,z}_{i:j} = \exp\left[\mathbf{w}^\top_{\sigma^{persist},d} \mathbf{f}^{d,s}_{\sigma^{persist}}(i, j, z)\right] \tag{63}$$

$$A^{d,s,z}_{u,v,i} = \exp\left[\mathbf{w}^\top_{\sigma^{transit},d} \mathbf{f}^{d,s}_{\sigma^{transit},u,v}(i, z)\right] \tag{64}$$

$$\pi^{d,s,z}_{u,i} = \exp\left[\mathbf{w}^\top_{\sigma^{init},d} \mathbf{f}^{d,s}_{\sigma^{init},u}(i, z)\right] \tag{65}$$

$$E^{d,s,z}_{u,i} = \exp\left[\mathbf{w}^\top_{\sigma^{end},d} \mathbf{f}^{d,s}_{\sigma^{end},u}(i, z)\right] \tag{66}$$

Denote by $\mathbf{F}^d_\sigma(\zeta, z)$ the global feature, which is the sum of all active features $\mathbf{f}^d_\sigma(z)$ at level d in the duration $[1, T]$ for a given assignment of ζ and a clique type σ. Recall that $\tau^d = \{i_k\}_{k=1}^m$ is the set of ending time indices (i.e. $e^d_{i_k} = 1$). The four feature types are given in Eqs. (67)–(70).

$$\mathbf{F}^{d,s}_{\sigma^{persist}}(\zeta, z) = \mathbf{f}^{d,s}_{\sigma^{persist}}(1, i_1, z) + \sum_{i_k \in \tau^d, k>1} \mathbf{f}^{d,s}_{\sigma^{persist}}(i_k + 1, i_{k+1}, z) \tag{67}$$

$$\mathbf{F}^{d,s}_{\sigma^{transit},u,v}(\zeta, z) = \sum_{i_k \notin \tau^{d-1}, i_k \in \tau^d} \mathbf{f}^{d,s}_{\sigma^{transit},u,v}(i_k, z) \tag{68}$$

$$\mathbf{F}^{d,s}_{\sigma^{init},u}(\zeta, z) = \mathbf{f}^{d,s}_{\sigma^{init},u,v}(1, z) + \sum_{i_k \in \tau^d} \mathbf{f}^{d,s}_{\sigma^{init},u,v}(i_k + 1, z) \tag{69}$$

$$\mathbf{F}^{d,s}_{\sigma^{end},u}(\zeta, z) = \sum_{i_k \in \tau^d} \mathbf{f}^{d,s}_{\sigma^{end},u,v}(i, z) \tag{70}$$

Substituting the global features into potentials in Eqs. (8), (9) we obtain the following log-linear model:

$$\Pr(\zeta \mid z) = \frac{1}{Z(z)} \exp\left[\sum_{c \in C} \mathbf{w}^{\top}_{\sigma^c} \mathbf{F}_{\sigma^c}(\zeta, z)\right] \tag{71}$$

where $C = \{persist, transit, init, exit\}$.

Again, for clarity of presentation we will drop the notion of z but implicitly assume that it is still in the each quantity.

6.2. ESS for state-persistence features

Recall from Section 6.1 that the feature function for the state-persistence $\mathbf{f}^{d,s}_{\sigma^{persist}}(i, j)$ is active only in the context where $\Pi^{d,s}_{i:j} \in \zeta$. Thus, Eq. (67) can be rewritten as

$$\mathbf{F}^{d,s}_{\sigma^{persist}}(\zeta) = \sum_{i \in [1,T]} \sum_{j \in [i,T]} \mathbf{f}^{d,s}_{\sigma^{persist}}(i, j) \delta\left[\Pi^{d,s}_{i:j} \in \zeta\right] \tag{72}$$

The indicator function in the RHS ensures that the feature $\mathbf{f}^{d,s}_{\sigma^{persist}}(i, j)$ is only active if there exists a symmetric Markov blanket $\Pi^{d,s}_{i:j}$ in the assignment of ζ. Consider the following expectation

$$\mathbb{E}\left[\mathbf{f}^{d,s}_{\sigma^{persist}}(i, j) \delta\left[\Pi^{d,s}_{i:j} \in \zeta\right]\right] = \sum_{\zeta} \Pr(\zeta) \mathbf{f}^{d,s}_{\sigma^{persist}}(i, j) \delta\left[\Pi^{d,s}_{i:j} \in \zeta\right] \tag{73}$$

$$= \frac{1}{Z} \sum_{\zeta} \Phi[\zeta] \mathbf{f}^{d,s}_{\sigma^{persist}}(i, j) \delta\left[\Pi^{d,s}_{i:j} \in \zeta\right] \tag{74}$$

Using the factorisation in Eq. (22) we can rewrite

$$\mathbb{E}\left[\mathbf{f}^{d,s}_{\sigma^{persist}}(i, j) \delta\left[\Pi^{d,s}_{i:j} \in \zeta\right]\right] = \frac{1}{Z} \sum_{\zeta} \Phi\left[\hat{\zeta}^{d,s}_{i:j}\right] \Phi\left[\underline{\hat{\zeta}}^{d,s}_{i:j}\right] R^{d,s}_{i:j} \mathbf{f}^{d,s}_{\sigma^{persist}}(i, j) \delta\left[\Pi^{d,s}_{i:j} \in \zeta\right] \tag{75}$$

Note that the elements inside the sum of the RHS are only non-zeros for those assignment of ζ that respect the persistent state $s^d_{i:j}$ and the factorisation in Eq. (22), i.e. $\zeta = (\zeta^{d,s}_{i:j}, \underline{\zeta}^{d,s}_{i:j}, \Pi^{d,s}_{i:j})$. Thus, the equation can be simplified to

$$\mathbb{E}\left[\mathbf{f}^{d,s}_{\sigma^{persist}}(i, j) \delta\left[\Pi^{d,s}_{i:j} \in \zeta\right]\right] = \frac{1}{Z} \sum_{\zeta^{d,s}_{i:j}} \sum_{\underline{\zeta}^{d,s}_{i:j}} \Phi\left[\hat{\zeta}^{d,s}_{i:j}\right] \Phi\left[\underline{\hat{\zeta}}^{d,s}_{i:j}\right] R^{d,s}_{i:j} \mathbf{f}^{d,s}_{\sigma^{persist}}(i, j) \tag{76}$$

$$= \frac{1}{Z} \Delta^{d,s}_{i:j} \Lambda^{d,s}_{i:j} R^{d,s}_{i:j} \mathbf{f}^{d,s}_{\sigma^{persist}}(i, j) \tag{77}$$

Using Eq. (72) we obtain the ESS for the state-persistence features

$$\mathbb{E}\left[F^{d,s}_k(\zeta)\right] = \sum_{i \in [1,T]} \sum_{j \in [i,T]} \mathbb{E}\left[\mathbf{f}^{d,s}_{\sigma^{persist}}(i, j) \delta\left[\Pi^{d,s}_{i:j} \in \zeta\right]\right]$$

$$= \frac{1}{Z} \sum_{i \in [1,T]} \sum_{j \in [i,T]} \Delta^{d,s}_{i:j} \Lambda^{d,s}_{i:j} R^{d,s}_{i:j} \mathbf{f}^{d,s}_{\sigma^{persist}}(i, j) \tag{78}$$

There are two special cases: (1) when $d = 1$, we do not sum over i, j but fix $i = 1, j = T$, and (2) when $d = D$ then we keep $j = i$.

6.3. ESS for transition features

Recall that in Section 6.1 we define $\mathbf{f}^{d,s}_{\sigma^{transit},u,v}(t)$ as a function that is active in the context $c^{transit} = \left[e^{d-1}_t = 0, e^d_t = 1\right]$, in which the child state u^d finishes its job at time t and transits to the child state v^d under the same parent s^{d-1} (that is s^{d-1} is still running). Thus Eq. (68) can be rewritten as

$$\mathbf{F}^{d,s}_{\sigma^{transit},u,v}(\zeta) = \sum_{t\in[1,T-1]} \mathbf{f}^{d,s}_{\sigma^{transit},u,v}(t)\delta\left[c^{transit} \in \zeta\right] \tag{79}$$

We now consider the following expectation

$$\mathbb{E}\left[\mathbf{f}^{d,s}_{\sigma^{transit},u,v}(t)\delta\left[c^{transit} \in \zeta\right]\right] = \sum_{\zeta} \Pr(\zeta)\mathbf{f}^{d,s}_{\sigma^{transit},u,v}(t)\delta\left[c^{transit} \in \zeta\right] \tag{80}$$

$$= \frac{1}{Z}\sum_{\zeta} \Phi[\zeta]\mathbf{f}^{d,s}_{\sigma^{transit},u,v}(t)\delta\left[c^{transit} \in \zeta\right] \tag{81}$$

Assume that the parent s starts at i. Since $e^d_t = 1$, the child v must starts at $t+1$ and ends some time later at $j \geq t+1$. We have the following decomposition of the configuration ζ that respects this assumption

$$\zeta = \left(\hat{\zeta}^{d-1,s}_{i:j}(v), \hat{\zeta}^{d-1,s}_{i:t}(u), \hat{\zeta}^{d,v}_{t+1:j}\right) \tag{82}$$

and the following factorisation of the joint potential

$$\Phi[\zeta] = \Phi\left[\hat{\zeta}^{d-1,s}_{i:j}(v)\right]\Phi\left[\hat{\zeta}^{d-1,s}_{i:t}(u)\right]\Phi\left[\hat{\zeta}^{d,v}_{t+1:j}\right]R^{d,v}_{t+1:j}A^{d,s}_{u,v,t} \tag{83}$$

The state persistent potential $R^{d,v}_{t+1:j}$ is enabled in the context $c = \left[e^d_t = 1, e^d_{t+1:j-1} = 0, e^d_j = 1\right]$ and the state transition potential $A^{d,s}_{u,v,t}$ in the context $c^{transit}$.

Substituting this factorisation into the RHS of Eq. (81) gives us

$$\frac{1}{Z}\sum_{i\in[1,t]}\sum_{j\in[t+1,T]}\sum_{\zeta^{d-1,s}_{i:t}(u)}\sum_{\zeta^{d-1,s}_{i:j}(v)}\sum_{\zeta^{d,v}_{t+1:j}} \Phi\left[\hat{\zeta}^{d-1,s}_{i:j}(v)\right]\Phi\left[\hat{\zeta}^{d-1,s}_{i:t}(u)\right]\Phi\left[\hat{\zeta}^{d,v}_{t+1:j}\right]R^{d,v}_{t+1:j}A^{d,s}_{u,v,t}\mathbf{f}^{d,s}_{\sigma^{transit},u,v}(t)$$

which can be simplified to

$$\frac{1}{Z}\sum_{i\in[1,t]}\sum_{j\in[t+1,T]} \lambda^{d-1,s}_{i:j}(v)\alpha^{d-1,s}_{i:t}(u)\hat{\Delta}^{d,v}_{t+1:j}A^{d,s}_{u,v,t}\mathbf{f}^{d,s}_{\sigma^{transit},u,v}(t) \tag{84}$$

Using Eqs. (79) and (84) we obtain the ESS for the state-transition features

$$\mathbb{E}\left[\mathbf{F}^{d,s}_{\sigma^{transit},u,v}(\zeta)\right] = \sum_{t\in[1,T-1]}\mathbb{E}\left[\mathbf{f}^{d,s}_{\sigma^{transit},u,v}(t)\delta[c^{transit} \in \zeta]\right]$$

$$= \frac{1}{Z}\sum_{t\in[1,T-1]} A^{d,s}_{u,v,t}\mathbf{f}^{d,s}_{\sigma^{transit},u,v}(t)\sum_{i\in[1,t]}\sum_{j\in[t+1,T]}\alpha^{d-1,s}_{i:t}(u)\lambda^{d-1,s}_{i:j}(v)\hat{\Delta}^{d,v}_{t+1:j} \tag{85}$$

When $d = 2$ we must fix $i = 1$ since $\alpha^{1,s}_{i:t}(u)$ and $\lambda^{1,s}_{i:j}(v)$ are only defined at $i = 1$.

6.4. ESS for initialisation features

Recall that in Section 6.1 we define $\mathbf{f}^{d,s}_{\sigma^{init},u}(i)$ as a function at level d that is triggered at time i when a parent s at level d initialises a child u at level $d+1$. In this event, the context $c^{init} = \left[e^d_{i-1} = 1\right]$ must be activated for $i > 1$. Thus, Eq. (69) can be rewritten as

$$\mathbf{F}^{d,s}_{\sigma^{init},u}(\zeta) = \sum_{i\in[1,T]} \mathbf{f}^{d,s}_{\sigma^{init},u}(i)\delta\left[c^{init} \in \zeta\right] \tag{86}$$

Now we consider the following feature expectation

$$\mathbb{E}\left[\mathbf{f}^{d,s}_{\sigma^{init},u}(i)\delta[c^{init} \in \zeta]\right] = \sum_{\zeta}\Pr(\zeta)\mathbf{f}^{d,s}_{\sigma^{init},u}(i)\delta\left[c^{init} \in \zeta\right]$$

$$= \frac{1}{Z}\sum_{\zeta}\Phi[\zeta]\mathbf{f}^{d,s}_{\sigma^{init},u}(i)\delta\left[c^{init} \in \zeta\right] \tag{87}$$

For each assignment of ζ that enables $\mathbf{f}^{d,s}_{\sigma^{init},u}(i)$, we have the following decomposition

$$\zeta = \left(\hat{\underline{\zeta}}^{d,s}_{i:j}(u), \hat{\zeta}^{d+1,u}_{i:j}\right) \tag{88}$$

where the context c^{init} activates the emission from s to u and the feature function $\mathbf{f}^{d,s}_{\sigma^{init},u}(i)$. Thus the joint potential $\Phi[\zeta]$ can be factorised as

$$\Phi[\zeta] = \Phi\left[\hat{\underline{\zeta}}^{d,s}_{i:j}(u)\right]\Phi\left[\hat{\zeta}^{d+1,u}_{i:j}\right]R^{d+1,u}_{i:j}\pi^{d,s}_{u,i} \tag{89}$$

Using this factorisation and noting that the elements within the summation in the RHS of Eq. (87) are only non-zeros with such assignments, we can simplify the RHS of Eq. (87) to

$$\begin{aligned}&\frac{1}{Z}\sum_{j\in[i,T]}\sum_{\hat{\underline{\zeta}}^{d,s}_{i:j}(u)}\sum_{\hat{\zeta}^{d+1,u}_{i:j}}\Phi\left[\hat{\underline{\zeta}}^{d,s}_{i:j}(u)\right]\Phi\left[\hat{\zeta}^{d+1,u}_{i:j}\right]R^{d+1,u}_{i:j}\pi^{d,s}_{u,i}\mathbf{f}^{d,s}_{\sigma^{init},u}(i)\\ &=\frac{1}{Z}\sum_{j\in[i,T]}\lambda^{d,s}_{i:j}(u)\hat{\Delta}^{d+1,u}_{i:j}\pi^{d,s}_{u,i}\mathbf{f}^{d,s}_{\sigma^{init},u}(i)\end{aligned} \tag{90}$$

The summation over $j \in [i, T]$ is due to the fact that we do not know this index.

Using Eqs. (86), (90) we obtain the ESS for the initialisation features

$$\begin{aligned}\mathbb{E}\left[\mathbf{F}^{d,s}_{\sigma^{init},u}(\zeta)\right] &= \sum_{i\in[1,T]}\mathbb{E}\left[\mathbf{f}^{d,s}_{\sigma^{init},u}(i)\delta[c^{init}\in\zeta]\right]\\ &= \frac{1}{Z}\sum_{i\in[1,T]}\pi^{d,s}_{u,i}\mathbf{f}^{d,s}_{\sigma^{init},u}(i)\sum_{j\in[i,T]}\lambda^{d,s}_{i:j}(u)\hat{\Delta}^{d+1,u}_{i:j}\end{aligned} \tag{91}$$

There are two special cases: (1) when $d = 1$, there must be no scanning of i but fix $i = 1$ since there is only a single initialisation at the beginning of the sequence, (2) when $d = D - 1$, we fix $j = i$ for $\hat{\Delta}^{D,u}_{i:j}$ is only defined at $i = j$.

6.5. ESS for ending features

Recall that in Section 6.1 we define $\mathbf{f}^{d,s}_{\sigma^{end},u}(j)$ as a function that is activated when a child u at level $d+1$ returns the control to its parent s at level d and time j. This event also enables the context $c^{end} = \left[e^d_j = 1\right]$. Thus Eq. (70) can be rewritten as

$$\mathbf{F}^{d,s}_{\sigma^{end},u}(\zeta) = \sum_{j\in[1,T]}\mathbf{f}^{d,s}_{\sigma^{end},u}(j)\delta\left[c^{end}\in\zeta\right] \tag{92}$$

Now we consider the following feature expectation

$$\begin{aligned}\mathbb{E}\left[\mathbf{f}^{d,s}_{\sigma^{end},u}(j)\delta[c^{end}\in\zeta]\right] &= \sum_{\zeta}\Pr(\zeta)\mathbf{f}^{d,s}_{\sigma^{end},u}(j)\delta\left[c^{end}\in\zeta\right]\\ &= \frac{1}{Z}\sum_{\zeta}\Phi[\zeta]\mathbf{f}^{d,s}_{\sigma^{end},u}(j)\delta\left[c^{end}\in\zeta\right]\end{aligned} \tag{93}$$

Assume that the state s starts at i and ends at j. For each assignment of ζ that enables $\mathbf{f}^{d,s}_{\sigma^{end},u}(j)$ and respects this assumption, we have the following decomposition

$$\zeta = \left(\hat{\underline{\zeta}}^{d,s}_{i:j}, \hat{\zeta}^{d,s}_{i:j}(u)\right) \tag{94}$$

This assignment has the context c^{end} that activates the ending of u. Thus the joint potential $\Phi[\zeta]$ can be factorised as

$$\Phi[\zeta] = \Phi\left[\hat{\underline{\zeta}}^{d,s}_{i:j}\right]\Phi\left[\hat{\zeta}^{d,s}_{i:j}(u)\right]R^{d,s}_{i:j}E^{d,s}_{u,j} \tag{95}$$

Substituting this factorisation into the summation of the RHS of Eq. (93) yields

$$\sum_{i\in[1,j]}\sum_{\hat{\underline{\zeta}}^{d,s}_{i:j}}\sum_{\hat{\zeta}^{d,s}_{i:j}(u)}\Phi\left[\hat{\underline{\zeta}}^{d,s}_{i:j}\right]\Phi\left[\hat{\zeta}^{d,s}_{i:j}(u)\right]R^{d,s}_{i:j}E^{d,s}_{u,j}\mathbf{f}^{d,s}_{\sigma^{end},u}(j) = \sum_{i\in[1,j]}\hat{\Lambda}^{d,s}_{i:j}\alpha^{d,s}_{i:j}(u)E^{d,s}_{u,j}\mathbf{f}^{d,s}_{\sigma^{end},u}(j) \tag{96}$$

Using Eqs. (92) and (96) we obtain the ESS for the exiting features

$$\mathbb{E}\left[\mathbf{F}^{d,s}_{\sigma^{end},u}(\zeta)\right] = \sum_{j\in[1,T]} \mathbb{E}\left[\mathbf{f}^{d,s}_{\sigma^{end},u}(j)\delta[e^d_{i-1}\in\zeta]\right]$$
$$= \frac{1}{Z}\sum_{j\in[1,T]} E^{d,s}_{u,j}\mathbf{f}^{d,s}_{\sigma^{end},u}(j)\sum_{i\in[1,j]}\hat{\Lambda}^{d,s}_{i:j}\alpha^{d,s}_{i:j}(u) \tag{97}$$

There is a special case: when $d=1$ there must be no scanning of i, j but fix $i=1,\ j=T$.

7. Partial labels in learning and inference

So far we have assumed that training data is fully labelled, and that testing data does not have any labels. In this section we extend the AIO to handle the cases in which these assumptions do not hold. Specifically, it may happen that the training data is not completely labelled, possibly due to lack of labelling resources. In this case, the learning algorithm should be robust enough to handle missing labels. On the other hand, during inference, we may partially obtain high quality labels from external sources. This requires the inference algorithm to be responsive to that data.

7.1. The constrained AIO algorithm

In this section we consider the general case when $\zeta=(\vartheta,h)$, where ϑ is the visible set labels, and h the hidden set. Since our HSCRF is also an exponential model it shares the same computation required for general CRFs (Eqs. (6) and (7)). We have to compute four quantities: the partial log-partition function $Z(\vartheta,z)$, the partition function $Z(z)$, the 'constrained' ESS $\mathbb{E}_{h|\vartheta,z}[\mathbf{F}(\vartheta,h,z)]$, and the 'free' ESS $\mathbb{E}_{\zeta|z}[\mathbf{F}(\zeta,z)]$. The partition function and the 'free' ESS has been computed in Sections 4 and 6, respectively. This section describes the other two quantities.

Let the set of visible labels be $\vartheta=(\widetilde{x},\widetilde{e})$ where $\widetilde{x}$ is the visible set of state variables and $\widetilde{e}$ is the visible set of ending indicators. The basic idea is that we have to modify procedures for computing the building blocks such as $\Delta^{d,s}_{i:j}$ and $\alpha^{d,s}_{i:j}(u)$, to address constraints imposed by the labels. For example, $\Delta^{d,s}_{i:j}$ implies that the state s at level d starts at i and persists till terminating at j. Then, if any labels (e.g. there is an $\widetilde{x}^d_k\neq s$ for $k\in[i,j]$) are seen, causing this assumption to be inconsistent, $\Delta^{d,s}_{i:j}$ will be zero. Therefore, in general, the computation of each building block is multiplied by an identity function that enforces the consistency between these labels and the required constraints for computation of that block. As an example, we consider the computation of $\Delta^{d,s}_{i:j}$ and $\alpha^{d,s}_{i:j}(u)$.

The symmetric inside mass $\Delta^{d,s}_{i:j}$ is consistent only if all of the following conditions are satisfied:

1. If there are state labels $\widetilde{x}^d_k$ at level d within the interval $[i,j]$, then $\widetilde{x}^d_k=s$,
2. If there is any label of ending indicator $\widetilde{e}^d_{i-1}$, then $\widetilde{e}^d_{i-1}=1$,
3. If there is any label of ending indicator $\widetilde{e}^d_k$ for some $k\in[i,j-1]$, then $\widetilde{e}^d_k=0$, and
4. If any ending indicator $\widetilde{e}^d_j$ is labelled, then $\widetilde{e}^d_j=1$.

These conditions are captured by using the following identity function:

$$\mathbb{I}\left[\Delta^{d,s}_{i:j}\right] = \delta\left[\widetilde{x}^d_{k\in[i,j]}=s\right]\delta\left[\widetilde{e}^d_{i-1}=1\right]\delta\left[\widetilde{e}^d_{k\in[i:j-1]}=0\right]\delta\left[\widetilde{e}^d_j=1\right] \tag{98}$$

When labels are observed, Eq. (40) is thus replaced by

$$\Delta^{d,s}_{i:j} = \mathbb{I}\left[\Delta^{d,s}_{i:j}\right]\left(\sum_{u\in S^{d+1}}\alpha^{d,s}_{i:j}(u)E^{d,s}_{u,j}\right) \tag{99}$$

Note that we do not need to explicitly enforce the state consistency in the summation over u since in the bottom-up and left-right computation, $\alpha^{d,s}_{i:j}(u)$ is already computed and contributes to the sum only if it is consistent.

Analogously, the asymmetric inside mass $\alpha^{d,s}_{i:j}(u)$ is consistent if all of the following conditions are satisfied:

1. The first three conditions for the symmetric inside mass $\Delta^{d,s}_{i:j}$ hold,
2. If the state at level d at time j is labelled, it must be u, and
3. If any ending indicator $\widetilde{e}^{d+1}_j$ is labelled, then $\widetilde{e}^{d+1}_j=1$.

These conditions are captured by the identity function

$$\mathbb{I}\left[\alpha^{d,s}_{i:j}(u)\right] = \delta\left[\widetilde{x}^d_{k\in[i,j]}=s\right]\delta\left[\widetilde{e}^d_{i-1}=1\right]\delta\left[\widetilde{e}^d_{k\in[i:j-1]}=0\right]\delta\left[\widetilde{x}^{d+1}_j=u\right]\delta\left[\widetilde{e}^{d+1}_j=1\right] \tag{100}$$

Thus Eq. (36) becomes

$$\alpha_{i:j}^{d,s}(u) = \mathbb{I}\left[\alpha_{i:j}^{d,s}(u)\right]\left(\sum_{k=i+1}^{j}\sum_{v\in S^{d+1}} \alpha_{i:k-1}^{d,s}(v)\hat{\Delta}_{k:j}^{d+1,u} A_{v,u,k-1}^{d,s} + \hat{\Delta}_{i:j}^{d+1,u}\pi_{u,i}^{d+1,s}\right) \tag{101}$$

Note that we do not need to explicitly enforce the state consistency in the summation over v and time consistency in the summation over k since in bottom-up computation, $\alpha_{i:j}^{d,s}(u)$ and $\Delta_{k:j}^{d+1,u}$ are already computed and contribute to the sum only if they are consistent. Finally, the constrained partition function $Z(\vartheta, z)$ is computed using Eq. (25) given that the inside mass is consistent with the observations.

Other building blocks, such as the symmetric outside mass $\Lambda_{i:j}^{d,s}$ and the asymmetric outside mass $\lambda_{i:j}^{d,s}(u)$, are computed in an analogous way. Since $\Lambda_{i:j}^{d,s}$ and $\Delta_{i:j}^{d,s}$ are complementary and they share (d, s, i, j), the same indicator function $\mathbb{I}\left[\Delta_{i:j}^{d,s}\right]$ can be applied. Similarly, the pair asymmetric inside mass $\alpha_{i:j}^{d,s}(u)$ and asymmetric outside mass $\lambda_{i:j}^{d,s}(u)$ are complementary and they share d, s, i, j, u, thus the same indicator function $\mathbb{I}\left[\alpha_{i:j}^{d,s}(u)\right]$ can be applied.

Once all constrained building blocks have been computed they can be used to calculate constrained ESS as in Section 6 without any further modifications. The only difference is that we need to replace the partition function $Z(z)$ by the constrained version $Z(\vartheta, z)$.

7.2. The constrained Viterbi algorithm

Recall that in the Generalised Viterbi Algorithm described in Section 5 we want to find the most probable configuration $\zeta^{MAP} = \arg\max_{\zeta} \Pr(\zeta \mid z)$. When some variables ϑ of ζ are labelled, it is not necessary to estimate them. The task is now to estimate the most probable configuration of the hidden variables h given the labels:

$$h^{MAP} = \arg\max_{h} \Pr(h \mid \vartheta, z) = \arg\max_{h} \Phi[h, \vartheta, z]$$

It turns out that the constrained MAP estimation is identical to the standard MAP except that we have to respect the labelled variables ϑ.

Since the Viterbi algorithm is just the max-product version of the AIO, the constrained Viterbi can be modified in the same manner as in the constrained AIO (Section 7.1). Specifically, for each auxiliary quantities such as $\Delta_{i:j}^{max,s}$ and $\alpha_{i:j}^{max,s}(u)$, we need to maintain a set of indicator functions that ensures the consistency with labels. Eqs. (98), (99) become

$$\begin{aligned}
\mathbb{I}\left[\Delta_{i:j}^{max,d,s}\right] &= \delta\left[\widetilde{x}_{k\in[i,j]}^{d} = s\right]\delta\left[\widetilde{e}_{i-1}^{d} = 1\right]\delta\left[\widetilde{e}_{k\in[i:j-1]}^{d} = 0\right]\delta\left[\widetilde{e}_{j}^{d} = 1\right] \\
\Delta_{i:j}^{max,d,s} &= \mathbb{I}\left[\Delta_{i:j}^{max,d,s}\right]\left(\max_{u\in S^{d+1}} \alpha_{i:j}^{max,d,s}(u) E_{u,j}^{d,s}\right)
\end{aligned} \tag{102}$$

Likewise, we have the modifications to Eq. (100) and Eq. (101), respectively.

$$\begin{aligned}
\mathbb{I}\left[\alpha_{i:j}^{max,d,s}(u)\right] &= \delta\left[\widetilde{x}_{k\in[i,j]}^{d} = s\right]\delta\left[\widetilde{e}_{i-1}^{d} = 1\right]\delta\left[\widetilde{e}_{k\in[i:j-1]}^{d} = 0\right]\delta\left[\widetilde{x}_{j}^{d+1} = u\right]\delta\left[\widetilde{e}_{j}^{d+1} = 1\right] \\
\alpha_{i:j}^{max,d,s}(u) &= \mathbb{I}\left[\alpha_{i:j}^{max,d,s}(u)\right]\max\left\{\max_{k\in[i+1,j]}\max_{v\in S^{d+1}} \alpha_{i:k-1}^{max,d,s}(v)\hat{\Delta}_{k:j}^{max,d+1,u} A_{v,u,k-1}^{d,s};\right. \\
&\qquad\qquad \left.\hat{\Delta}_{i:j}^{max,d+1,u}\pi_{u,i}^{d+1,s}\right\}
\end{aligned} \tag{103}$$

Other tasks in the Viterbi algorithm including bookkeeping and backtracking are identical to those described in Section 5.

7.3. Complexity analysis

The complexity of the constrained AIO and constrained Viterbi has an upper bound of $\mathcal{O}(T^3)$, when no labels are given. It also has a lower bound of $\mathcal{O}(T)$ when all ending indicators are known and the model reduces to the standard tree-structured graphical model. In general, the complexity decreases as more labels are available, and we can expect a sub-cubic time behaviour.

Learning requires both the constrained ESSes and free ESSes to be computed. Regardless of labels, the free ESSes still require cubic time. Thus with less labels, the overall computation will increase slightly.

8. Numerical scaling

In previous sections, we have derived AIO-based inference and learning algorithms for both unconstrained and constrained models. The quantities computed by these algorithms like the inside/outside masses often involve summation over exponentially many positive potentials. The potentials, when estimated from data, are not upper-bounded, causing the magnitude of the masses to increase exponentially fast in the sequence length T. The magnitude goes beyond the numerical capacity of most machines for moderate T. In this section we present a scaling method to reduce this *numerical overflow* problem.

8.1. Scaling symmetric/asymmetric inside masses

Let us revisit Eq. (40). If we scale down the asymmetric inside mass $\alpha_{i:j}^{d,s}(u)$ by a factor $\kappa_j > 1$, i.e.

$$\alpha_{i:j}'^{d,s}(u) \leftarrow \frac{\alpha_{i:j}^{d,s}(u)}{\kappa_j} \tag{104}$$

then the symmetric inside mass $\Delta_{i:j}^{d,s}$ is also scaled down by the same factor. Similarly, as we can see from Eq. (36) that

$$\alpha_{i:j}^{d,s}(u) = \sum_{t=i+1}^{j} \sum_{v \in S^{d+1}} \alpha_{i:t-1}^{d,s}(v) \hat{\Delta}_{t:j}^{d+1,u} A_{v,u,t-1}^{d,s} + \hat{\Delta}_{i:j}^{d+1,u} \pi_{u,i}^{d,s}$$

where $\hat{\Delta}_{t:j}^{d+1,u} = \Delta_{t:j}^{d+1,u} R_{t:j}^{d+1,u}$, if $\Delta_{t:j}^{d+1,u}$ for $t \in [1, j]$ is reduced by κ_j, then $\alpha_{i:j}^{d,s}$ is also reduced by the same factor. In addition, using the set of recursive relations in Eqs. (36), (40), any reduction at the bottom level of $\Delta_{j:j}^{D,s}$ will result in the reduction of the symmetric inside mass $\Delta_{i:j}^{d,s}$ and of the asymmetric inside mass $\alpha_{i:j}^{d,s}(u)$, for $d < D$, by the same factor.

Suppose $\Delta_{i:i}^{D,s}$ is reduced by a factor of $\kappa_i > 1$ for all $i \in [1, j]$, the quantities $\Delta_{1:j}^{d,s}$ and $\alpha_{1:j}^{d,s}(u)$ will be reduced by a factor of $\prod_{i=1}^{j} \kappa_i$. That is

$$\hat{\Delta}_{1:j}'^{d,s} \leftarrow \frac{\hat{\Delta}_{1:j}^{d,s}}{\prod_{i=1}^{j} \kappa_i} \tag{105}$$

$$\alpha_{1:j}'^{d,s}(u) \leftarrow \frac{\alpha_{1:j}^{d,s}(u)}{\prod_{i=1}^{j} \kappa_i} \tag{106}$$

It follows immediately from Eq. (25) that the partition function is scaled down by a factor of $\prod_{i=1}^{T} \kappa_i$

$$Z' = \sum_{s \in S^1} \hat{\Delta}_{1:T}'^{1,s} = \frac{Z}{\prod_{j=1}^{T} \kappa_j} \tag{107}$$

where $\hat{\Delta}_{1:T}'^{1,s} = \Delta_{1:T}'^{1,s} B_{1:T}^{1,s}$. Clearly, we should deal with the log of this quantity to avoid numerical overflow. Thus, the log-partition function can be computed as

$$\log(Z) = \log \sum_{s \in S^1} \hat{\Delta}_{1:T}'^{1,s} + \sum_{j=1}^{T} \log \kappa_j \tag{108}$$

where $\Delta_{1:T}'^{1,s}$ has been scaled appropriately.

One question is how to choose the set of meaningful scaling factors $\{\kappa_j\}_1^T$. The simplest way is to choose a relatively large number for all scaling factors but making the right choice is not straightforward. Here we describe a more natural way to do so. Assume that we have chosen all the scaling factors $\{\kappa_i\}_1^{j-1}$. Using the original Eqs. (36), (37), and (38), where all the sub-components have been scaled appropriately, we compute the *partially-scaled* inside mass $\Delta_{i:j}''^{d,s}$ for $d \in [2, D]$ and asymmetric inside mass $\alpha_{i:j}''^{d,s}(u)$, for $d \in [1, D-1]$ and $i \in [1, j]$. Then the scaling factor at time j is computed as

$$\kappa_j = \sum_{s,u} \alpha_{1:j}''^{1,s}(u) \tag{109}$$

The next step is to rescale all the partially-scaled variables:

$$\alpha'^{d,s}_{i:j}(u) \leftarrow \frac{\alpha''^{d,s}_{i:j}(u)}{\kappa_j} \text{ for } s \in S^d, d \in [1, D-1] \tag{110}$$

$$\Delta'^{d,s}_{i:j} \leftarrow \frac{\Delta''^{d,s}_{i:j}}{\kappa_j} \text{ for } s \in S^d, d \in [2, D-1] \tag{111}$$

$$\Delta'^{D,s}_{j:j} \leftarrow \frac{\Delta''^{D,s}_{j:j}}{\kappa_j} \text{ for } s \in S^D \tag{112}$$

where $i \in [1, j]$.

8.2. Scaling symmetric/asymmetric outside masses

In a similar fashion we can work out the set of factors from the derivation of symmetric/asymmetric outside masses since these masses solely depend on the inside masses as building blocks. In other words, after scaling the inside masses we can compute the scaled outside masses directly, using the same set of equations described in Section 4.3.

The algorithm is summarised in Algorithm 7. Note that the order of performing the loops in this case is different from that in Algorithm 1.

Algorithm 7 Scaling algorithm to avoid numerical overflow.

Input: D, T and all the contextual potentials.
Output: Scaled inside/asymmetric inside masses, outside/asymmetric outside masses.

For $j = 1, 2, .., T$
  Compute $\alpha^{d,s}_{1:j}(u)$, $d \in [1, D-1]$ using Eqs. (36), (37) and (38)
  Compute κ_j using Eq. (109)
  Rescale $\alpha^{1,s}_{1:j}(u)$ using Eq. (110)
  For $i = 1, 2, .., j$
    For $d = 2, 3, .., D-1$
      Rescale $\alpha^{d,s}_{i:j}(u)$ using Eq. (110)
      Rescale $\Delta^{d,s}_{i:j}$ using Eq. (111)
    EndFor
  EndFor
  Rescale $\Delta^{D,s}_{j:j}$ using Eq. (112)
EndFor
Compute true log-partition function using Eq. (108).
Compute the outside/asymmetric outside masses using the
  scaled inside/asymmetric inside masses instead of the original
  inside/asymmetric inside in Eqs. (43) and (47).

9. Applications

In this section we present experimental results to demonstrate the capacity of the proposed HSCRFs in two applications: *activity recognition* and *shallow parsing*.

9.1. Recognising indoor activities

In this experiment, we evaluate the HSCRFs with a relatively small dataset from the domain of indoor video surveillance. The task is to recognise indoor trajectories and activities of a person from his noisy positions extracted from video. The data was captured in [21], and was subsequently used to evaluate DCRFs in [22]. The raw data consists of 90 sequences of noisy coordinates. Each time step is manually annotated by two labels: the complex and primitive activities. There are three complex activities (*preparing-short-meal*, *having-snack* and *preparing-normal-meal*); and 12 primitive activities listed in Table 8. As in [22], the data is split into two sets of equal size for training and testing, respectively.

We assume that state-specific features such as initialisation, transition and exiting are indicator functions. For the data-associations (i.e. embedded in state-persistence potentials) at the bottom level, we use the same feature set as in [22], which are: the (X, Y) coordinates, the X & Y velocities, and the speed. At the second level during duration of a state, we use average velocities and a vector of positions visited within that duration. To encode the duration into the state-persistence potentials, we employ the sufficient statistics of the *gamma* distribution as features $f_k(s, \Delta t) = \mathbb{I}(s)\log(\Delta t)$ and $f_{k+1}(s, \Delta t) = \mathbb{I}(s)(\Delta t)$.

Table 8
Primitive activities [21].

No.	Activity	No.	Activity
1	Door→Cupboard	7	Fridge→TV chair
2	Cupboard→Fridge	8	TV chair→Door
3	Fridge→Dining chair	9	Fridge→Stove
4	Dining chair→Door	10	Stove→Dining chair
5	Door→TV chair	11	Fridge→Door
6	TV chair→Cupboard	12	Dining chair→Fridge

Table 9
Accuracy (%) for fully observed labels (left), and 50% partially observed (PO) labels (right).

Alg.	$d=2$	$d=3$	Alg.	$d=2$	$d=3$
HSCRF	100	93.9	PO-HSCRF	80.2	90.4
DCRF	96.5	89.7	PO-CRF	–	83.5
flat-CRF	–	82.6	–	–	–

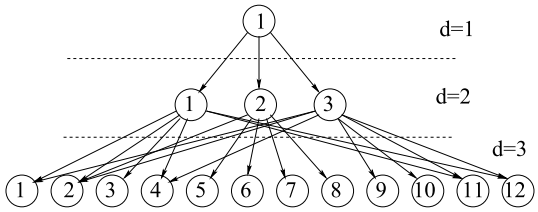

Fig. 9. The topology learned from data.

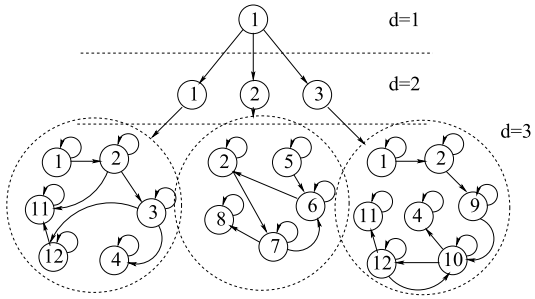

Fig. 10. The state transition model learned from data. Primitive states are duplicated for clarity only. They are shared among complex states.

For learning, labels for each sequence are provided fully for the case of fully observed state data, and partially for the case of missing state data. For testing, at each level d and time t we count an error if the predicted state is not the same as the ground-truth.

9.1.1. Fully supervised learning

Firstly, we examine the fully observed case where the HSCRF is compared against the DCRF at both data levels, and against the flat-CRF at bottom level. Table 9 (the left half) shows that (a) both the multilevel models significantly outperform the flat model and (b) the HSCRF outperforms the DCRF.

We also test the ability of the model to learn the hierarchical topology and state transitions. Using log-linear parameterisation described in Section 6.1, the parent–child relationship in the topology and the state-transitions are captured in the corresponding parameters. Only positive parameters are used. This is because state features are non-negative, so negative parameters mean the probabilities of these transitions are very small (due to the exponential), compared to the positive ones. For the transition at the second level (the complex activity level), we obtain all negative entries. This clearly matches the training data, in which each sequence already belongs to one of three complex activities. With this method, we are able to construct the correct hierarchical topology as in Fig. 9. The state transition model is presented in Fig. 10. There is only one wrong transition, from state 12 to state 10, which is not presented in the training data. The rest is correct.

9.1.2. Partially supervised learning

Next we consider partially-supervised learning in that about 50% of start/end times of a segment and segment labels are observed at the second level. All ending indicators are known at the bottom level. The results are reported in Table 9 (the

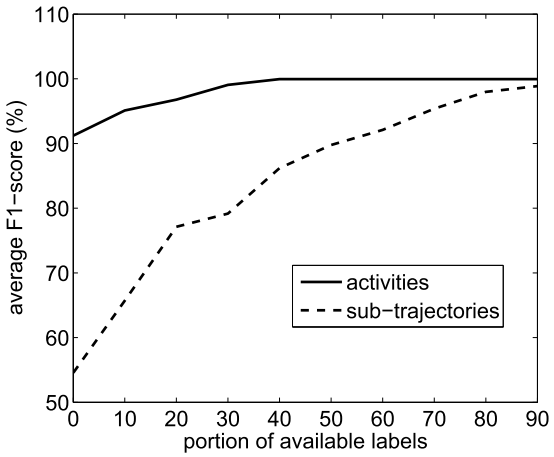

Fig. 11. Performance of the constrained max-product algorithm as a function of available information on label/start/end time.

right half). As can be seen, although only 50% of the state labels and state start/end times are observed, the model learned is still performing well with accuracy of 80.2% and 90.4% at levels 2 and 3, respectively.

9.1.3. Prediction with partial labelling

We now consider the issue of using partial observed labels during decoding to improve prediction accuracy of poorly estimated models. We extract the parameters from the 10th iteration of the fully observed data case. The labels are provided at random time indexes. Fig. 11a shows the decoding accuracy as a function of available state labels. It is interesting to observe that a moderate amount of observed labels (e.g. 20–40%) causes the accuracy rate to go up considerably.

9.2. POS tagging and noun-phrase chunking

In this experiment we apply the HSCRF to the task of noun-phrase chunking. The data is from the CoNLL-2000 shared task [5], in which 8,926 English sentences from the Wall Street Journal corpus are used for training and 2,012 sentences are for testing. Each word in a pre-processed sentence is labelled by two labels: the part-of-speech (POS) and the noun-phrase (NP). There are 48 POS different labels and 3 NP labels (B-NP for beginning of a noun-phrase, I-NP for inside a noun-phrase or O for others). Each noun-phrase generally has more than one word. To reduce the computational burden, we reduce the POS tag-set to 5 groups: *noun, verb, adjective, adverb* and *others*. Since in our HSCRFs we do not have to explicitly indicate which node is at the beginning of a segment, the NP label set can be reduced further into NP for noun-phrase, and O for anything else.

The POS tags are actually the output of the Brill's tagger [23], while the NPs are manually labelled. We extract raw features from the text in the way similar to that in [8]. However, we consider only a limited vocabulary extracted from the training data in that we only select words with more than 3 occurrences. This reduces the vocabulary and the feature size significantly. We also make use of bi-grams with similar selection criteria. Furthermore, we use the contextual window of 5 instead of 7 as in [8]. This setting gives rise to about 32K raw features. The model feature is factorised as $f(x_c, z) = \mathbb{I}(x_c)g_c(z)$, where $\mathbb{I}(x_c)$ is a binary function on the assignment of the clique variables x_c, and $g_c(z)$ are the raw features.

We build an HSCRF topology of 3 levels where the root is just a dummy node, the second level has 2 NP states and the bottom level has 5 POS states. For comparison, we implement a DCRF, a simple sequential CRF (SCRF), and a semi-Markov CRF (SemiCRF) [13]. The DCRF has grid structure of depth 2, one for modelling the NP process and another for the POS process. Since the state spaces are relatively small, we are able to run exact inference in the DCRF by collapsing both the NP and POS state spaces to a combined state space of size $3 \times 5 = 15$. The SCRF and SemiCRF model only the NP process, taking the POS tags as input. The raw feature set used in the DCRF is identical to those in our HSCRF. However, the set shared by the SCRF and the SemiCRF is a little more elaborate since it takes the POS tags into account [8].

Although both the HSCRF and the SemiCRF are capable of modelling arbitrary segment duration, we use a simple exponential distribution as it can be processed sequentially and thus is very efficient. For learning, we use a simple online stochastic gradient ascent method since it has been shown to work relatively well and fast in CRFs [24]. At test time, as the SCRF and the SemiCRF are able to use the Brill's POS tags as input, it is not fair for the DCRF and HSCRF to predict those labels during inference. Instead, we also give the POS tags to the DCRF and HSCRF and perform constrained inference to predict *only* the NP labels. This boosts the performance of the two multi-level models significantly.

The performance of these models is depicted in Fig. 12 and we are interested in only the prediction of the noun-phrases since this data has Brill's POS tags. Without Brill's POS tags given at test time, both the HSCRF and the DCRF perform worse than the SCRF trained on POS tags. This is not surprising because the Brill's POS tags are always given in the case of SCRF. However, with POS tags given at test time, the HSCRF consistently works better than all other models. In particular, our

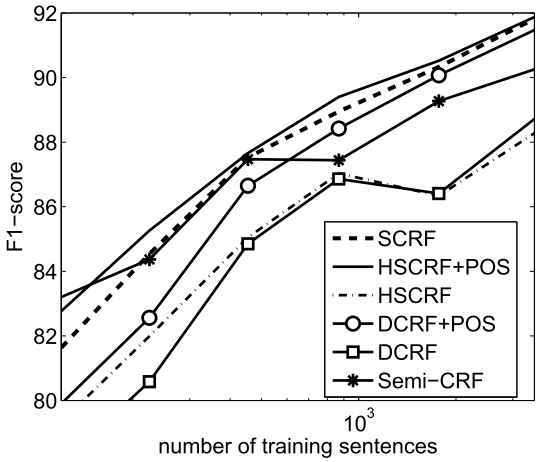

Fig. 12. Performance of various models on Conll2000 noun-phrase chunking. HSCRF + POS and DCRF + POS mean HSCRF and DCRF with POS given at test time, respectively.

HSCRF + POS is significantly better than the SCRF (with POS included as features) when training data is small. There is a subtle but important difference here: The HSCRF was trained *without* knowledge of POS availability at test time. In contrast, SCRF was trained with this knowledge so that good POS-based features were used.

The DCRF does worse than the SCRF, even with POS tags given. This does not share the observation made in [8]. However, we use a much smaller POS tag set than [8] does. Our explanation is that the SCRF is able to make use of wider context of the given POS tags (here, within the window of 5 tags) than the DCRF (limited to 1 POS tag per NP chunk). The SemiCRF, although in theory it is more expressive than the SCRF, does not show any advantage under current setting. Recall that the SemiCRF is a special case of HSCRF in that the POS level is not modelled, it is possible to conclude that joint modelling of NP and POS levels is important.

More formally, let us look at the difference between the flat setting of SCRF and Semi-CRF and the multi-level setting of DCRF and HSCRF. Let $x = (x_{np}, x_{pos})$. Essentially, we are about to model the distribution $\Pr(x \mid z) = \Pr(x_{np} \mid x_{pos}, z) \Pr(x_{pos} \mid z)$ in the multi-level models while we ignore the $\Pr(x_{pos} \mid z)$ in the flat models. During test time of the multi-level models, we predict only the x_{np} by finding the maximiser of $\Pr(x_{np} \mid x_{pos}, z)$. The POS model $\Pr(x_{pos} \mid z)$ seems to be a waste because we do not make use of it at test time. However, $\Pr(x_{pos} \mid z)$ does give extra information about the joint distribution $\Pr(x \mid z)$, that is, modelling the POS process may help to get smoother estimate of the NP distribution.

10. Related work

Hierarchical modelling of stochastic processes can be largely categorised as either graphical models extending the flat hidden (semi-)Markov models (HMM/HsMM) (e.g., the layered HMM [6], the abstract HMM [25], hierarchical HMM (HHMM) [2,15], DBN [26]), grammar-based models (e.g., PCFG [27]), or deterministic models [28,29]. These models are all directed while HSCRF is undirected.

Higher-order extensions to the linear-chain CRFs have been developed recently [30,31]. These methods exploit sparsity in the state transition for efficient inference, but they are shallow models. Development in deeper structures include dynamic CRFs (DCRF) [8], hierarchical CRFs [9,10], and stacked CRFs [11]. The main difference between HSCRF and these CRF variants is that *the hierarchical topology of HSCRF is dynamically inferred from data*, unlike the others, where the topology is pre-defined by users. In term of inference complexity, DCRFs are not tractable in large-state settings. The hierarchical CRFs, on the other hand, are tractable but assume fixed tree structures, and therefore are not flexible to adapt to complex data. For example, the noun-phrase chunking problem does not assume prior tree structures. Rather, if such a structure exists, it can only be discovered after the model has been successfully built and learned.

Our HSCRFs deal with the inference problem of DCRFs by limiting to recursive processes, and thus obtaining efficient inference via dynamic programming in the Inside–Outside family of algorithms. Furthermore, it generalises the SemiCRFs to model multilevel of semantics. It also addresses partial labels by introducing appropriate constraints to the Inside–Outside algorithms.

As the HHMMs are special case of PCFG with bounded depth, HSCRFs are also special case of the conditional probabilistic context-free grammar (C-PCFG) (e.g. see [32,33]). Like HSCRFs, C-PCFG requires cubic time in sequence length to parse a sentence. However, the context-free grammar does not limit the depth of semantic hierarchy, thus making it unnecessarily difficult to map many hierarchical problems into its form. Secondly, it lacks a graphical model representation and thus does not enjoy the rich set of approximate inference techniques available in graphical models.

The AIO algorithm presented in Section 4 is inspired from the AIO algorithm in HHMMs [15,2]. However, due to the log-linear parameterisation, there are no probabilistic interpretations of the inside and outside masses.

The idea of numerical scaling presented in Section 8 can be traced back to the Pearl's message-passing procedure [20, 34]. In our AIO algorithms, the inside masses play the role of the inside-out messages. In Pearl's method, we reduce the messages' magnitude by normalising them at each step. The overflow problem is opposite to the underflow in directed counterparts. A similar idea has been proposed in [15] for HHMMs.

The graphical model-like dynamic representation of the HSCRF appears similar to the DBN representation of the HHMMs in [17], and somewhat resembles a dynamic factor graph [35,36, Chap 9]. However, it is not exactly the standard graphical model because the contextual cliques in HSCRFs are not fixed during inference. This makes it difficult to apply approximate inference methods such as Loopy Belief Propagation (LBP) and Gibbs sampling, which are designed for fixed cliques. The Gibbs sampling can be applied to a special arrangement as in [37], but convergence is not guaranteed within limited time and our preliminary experiments have indicated no advantage compared to exact inference.

11. Conclusions

In this paper, we have presented a novel model called Hierarchical Semi-Markov Conditional Random Field which extends the standard CRFs to incorporate hierarchical and multilevel semantics. We have developed a graphical model-like dynamic representation of the HSCRF. We have derived efficient algorithms for learning and inference, especially the ability to learn and inference with partially given labels. We have demonstrated the capacity of the HSCRFs on home video surveillance data and the shallow parsing of English text, in which the hierarchical information inherent in the context helps to increase the recognition.

In future work we plan to attack the computational bottleneck in large-scale settings. Although the AIO family has cubic time complexity, it is still expensive in large-scale application, especially those with long sequences. It is therefore desirable to introduce approximation methods that can provide speed/quality trade-offs. Our early work using Rao-Blackwellised Gibbs sampling shows promising results [37]. We also need to make a choice between pre-computing all the potentials prior to inference and learning, and computing them on-the-fly. The first choice requires $\mathcal{O}(DK^3T^2)$ space, which is very significant with typical real-world problems, even with today's computing power. The second choice, however, will slow the inference and learning very significantly due to repeated computation at every step of the AIO algorithm. Finally, it is interesting to see how good the HSCRFs can be an approximation to general multilevel processes, which are not necessarily recursive (e.g., HSCRF as an approximation to DCRFs). This is important because HSCRFs are tractable while DCRFs are generally not.

Acknowledgement

This work is partially supported by the Telstra–Deakin Centre of Excellence in Big Data and Machine Learning.

Appendix A. Proofs

In this appendix we give detailed proofs of propositions stated in the main text.

A.1. Proof of Propositions 1 and 2

Before proving Proposition 1 and 2 let us introduce a lemma.

Lemma 1. *Given a distribution of the form* $\Pr(x) \propto \Phi[x]$ *and* $x = (x_a, x_s, x_b)$*, if there exists a factorisation*

$$\Phi[x] = \Phi[x_a, x_s]\Phi[x_s]\Phi[x_s, x_b] \tag{A.1}$$

then x_a *and* x_b *are conditionally independent given* x_s*.*

Proof. We want to prove that

$$\Pr(x_a, x_b \mid x_s) = \Pr(x_a \mid x_s)\Pr(x_b \mid x_s) \tag{A.2}$$

Since $\Pr(x_a, x_b \mid x_s) = \Pr(x_a, x_b, x_s)/\sum_{x_a,x_b}\Pr(x_a, x_b, x_s)$, the LHS of Eq. (A.2) becomes

$$\begin{aligned}\Pr(x_a, x_b \mid x_s) &= \frac{\Phi[x_a, x_s]\Phi[x_s]\Phi[x_s, x_b]}{\sum_{x_a,x_b}\Phi[x_a, x_s]\Phi[x_s]\Phi[x_s, x_b]} \\ &= \frac{\Phi[x_a, x_s]}{\sum_{x_a}\Phi[x_a, x_s]}\frac{\Phi[x_s, x_b]}{\sum_{x_b}\Phi[x_s, x_b]}\end{aligned} \tag{A.3}$$

where we have used the following fact

$$\sum_{x_a,x_b}\Phi[x_a, x_s]\Phi[x_s]\Phi[x_s, x_b] = \Phi[x_s]\Big(\sum_{x_a}\Phi[x_a, x_s]\Big)\Big(\sum_{x_b}\Phi[x_s, x_b]\Big) \tag{A.4}$$

To prove $\Pr(x_a \mid x_s) = \Phi[x_a, x_s] / \sum_{x_a} \Phi[x_a, x_s]$, we need only to show $\Pr(x_a \mid x_s) \propto \Phi[x_a, x_s]$ since the normalisation over x_a is due to $\sum_{x_a} \Pr(x_a|x_s) = 1$. Using the Bayes rule, we have

$$\begin{aligned}\Pr(x_a \mid x_s) \propto \Pr(x_a, x_s) &= \sum_{x_b} \Pr(x_a, x_s, x_b) \\ &\propto \Phi[x_a, x_s]\Phi[x_s] \sum_{x_b} \Phi[x_s, x_b] \\ &\propto \Phi[x_a, x_s] \end{aligned} \tag{A.5}$$

where we have ignored all the factors that do not depend on x_a.

A similar proof gives $\Pr(x_b \mid x_s) = \Phi[x_s, x_b] / \sum_{x_b} \Phi[x_s, x_b]$. Combining this result and Eq. (A.5) with Eq. (A.3) gives us Eq. (A.2). This completes the proof. □

In fact, x_s acts as a separator between x_a and x_b. In standard Markov networks there are no paths from x_a to x_b that do not go through x_s. Now we proceed to proving Propositions 1 and 2.

Given the symmetric Markov blanket $\Pi^{d,s}_{i:j}$, there are no potentials that are associated with variables belonging to both $\zeta^{d,s}_{i:j}$ and $\underline{\zeta}^{d,s}_{i:j}$. The blanket completely separates the $\zeta^{d,s}_{i:j}$ and $\underline{\zeta}^{d,s}_{i:j}$. Therefore, Lemma 1 ensures the conditional independence between $\zeta^{d,s}_{i:j}$ and $\underline{\zeta}^{d,s}_{i:j}$.

Similarly, the asymmetric Markov blanket $\Gamma^{d,s}_{i:j}(u)$ separates $\zeta^{d,s}_{i:j}(u)$ and $\underline{\zeta}^{d,s}_{i:j}(u)$ and thus these two variable sets are conditionally independent due to Lemma 1. □

A.2. Proof of Proposition 3

Here we want to derive Eqs. (28), (29) and (30). With the same conditions as in Lemma 1, in Eq. (A.5) we have shown that $\Pr(x_a \mid x_s) \propto \Phi[x_a, x_s]$. Similarly, this extends to

$$\Pr\left(\zeta^{d,s}_{i:j} \mid \Pi^{d,s}_{i:j}\right) \propto \Phi\left[\zeta^{d,s}_{i:j}, \Pi^{d,s}_{i:j}\right] = \Phi\left[\hat{\zeta}^{d,s}_{i:j}\right]$$

which is equivalent to

$$\Pr\left(\zeta^{d,s}_{i:j} \mid \Pi^{d,s}_{i:j}\right) = \frac{1}{\sum_{\zeta^{d,s}_{i:j}} \Phi\left[\hat{\zeta}^{d,s}_{i:j}\right]} \Phi\left[\hat{\zeta}^{d,s}_{i:j}\right] = \frac{1}{\Delta^{d,s}_{i:j}} \Phi\left[\hat{\zeta}^{d,s}_{i:j}\right]$$

The last equation follows from the definition of the symmetric inside mass in Eq. (23). Similar procedure will yield Eq. (29).

To prove Eq. (30), notice the Eq. (19) that says

$$\Pr(\zeta) = \Pr\left(\Pi^{d,s}_{i:j}\right) \Pr\left(\zeta^{d,s}_{i:j}] \mid \Pi^{d,s}_{i:j}\right) \Pr\left(\underline{\zeta}^{d,s}_{i:j} \mid \Pi^{d,s}_{i:j}\right) \tag{A.6}$$

or equivalently

$$\Pr(\Pi^{d,s}_{i:j}) = \Pr(\zeta) \frac{1}{\Pr\left(\zeta^{d,s}_{i:j} \mid \Pi^{d,s}_{i:j}\right)} \frac{1}{\Pr\left(\underline{\zeta}^{d,s}_{i:j} \mid \Pi^{d,s}_{i:j}\right)} \tag{A.7}$$

$$\propto \Phi[\zeta] \frac{\Delta^{d,s}_{i:j}}{\Phi[\hat{\zeta}^{d,s}_{i:j}]} \frac{\Lambda^{d,s}_{i:j}}{\Phi[\hat{\underline{\zeta}}^{d,s}_{i:j}]} \tag{A.8}$$

$$= \Phi[\hat{\zeta}^{d,s}_{i:j}] R^{d,s}_{i:j} \Phi[\hat{\underline{\zeta}}^{d,s}_{i:j}] \frac{\Delta^{d,s}_{i:j}}{\Phi[\hat{\zeta}^{d,s}_{i:j}]} \frac{\Lambda^{d,s}_{i:j}}{\Phi[\hat{\underline{\zeta}}^{d,s}_{i:j}]} \tag{A.9}$$

$$= \Delta^{d,s}_{i:j} R^{d,s}_{i:j} \Lambda^{d,s}_{i:j} \tag{A.10}$$

In the proof proceeding, we have made use of the relation in Eq. (22). This completes the proof. □

Appendix B. Computing state marginals

We are interested in computing the marginals of state variables $\Pr\left(x^d_t\right)$. We have

$$\begin{aligned}\Pr\left(x^d_t\right) &= \sum_{\zeta \setminus x^d_t} \Pr\left(x^d_t, \zeta \setminus x^d_t\right) = \sum_{\zeta} \Pr(\zeta) \delta\left[x^d_t \in \zeta\right] \\ &= \frac{1}{Z} \sum_{\zeta} \Phi[\zeta] \delta\left[x^d_t \in \zeta\right] \end{aligned} \tag{B.1}$$

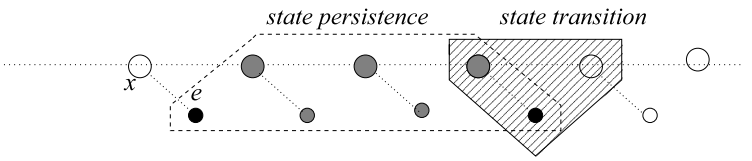

Fig. C.13. The SemiCRFs in our contextual clique framework.

Let $s = x_t^d$ and assume that the state s starts at i and end at j, and $t \in [i, j]$. For each configuration ζ that respects this assumption, we have the factorisation of Eq. (22) that says

$$\Phi[\zeta] = \Phi\left[\hat{\zeta}_{i:j}^{d,s}\right] \Phi\left[\hat{\zeta}_{\underline{i}:j}^{d,s}\right] R_{i:j}^{d,s} \tag{B.2}$$

Then Eq. (B.1) becomes

$$\begin{aligned} \Pr(x_t^d = s) &= \frac{1}{Z} \sum_{\zeta} \Phi\left[\hat{\zeta}_{i:j}^{d,s}\right] \Phi\left[\hat{\zeta}_{\underline{i}:j}^{d,s}\right] R_{i:j}^{d,s} \delta\left[t \in [i, j]\right] \\ &= \frac{1}{Z} \sum_{i \in [1,t]} \sum_{j \in [t,T]} \Delta_{i:j}^{d,s} \Lambda_{i:j}^{d,s} R_{i:j}^{d,s} \end{aligned} \tag{B.3}$$

The summing over i and j is due to the fact that we do not know these indices.

There are two special cases, (1) when $d = 1$ we cannot scan the left and right indices, the marginals are simply

$$\Pr(x_t^1 = s) = \frac{1}{Z} \hat{\Delta}_{1:T}^{1,s} \tag{B.4}$$

since $\Lambda_{1:T}^{1,s} = 1$ for all $s \in S^1$; and (2) when $d = D$, the start and end times must be the same ($i = j$), thus

$$\Pr(x_t^D = s) = \frac{1}{Z} \hat{\Lambda}_{t:t}^{D,s} \tag{B.5}$$

since $\Delta_{t:t}^{D,s} = 1$ for all $t \in [1, T]$ and $s \in S^D$.

Since $\sum_{s \in S^d} \Pr\left(x_t^d = s\right) = 1$, it follows from Eq. (B.3) that

$$Z = \sum_{s \in S^d} \sum_{i \in [1,t]} \sum_{j \in [t,T]} \Delta_{i:j}^{d,s} \Lambda_{i:j}^{d,s} R_{i:j}^{d,s} \tag{B.6}$$

This turns out to be the most general way of computing the partition function. Some special cases have been shown earlier. For example, when $d = 1$, $i = 1$ and $j = T$, Eq. (B.6) becomes Eq. (25) since $\Lambda_{1:T}^{1,s} = 1$. Similarly, when $d = D$, $i = j = t$, Eq. (B.6) recovers Eq. (26) since $\Delta_{i:i}^{D,s} = 1$.

Appendix C. Semi-Markov CRFs as a special case

In this appendix we show how to convert a semi-Markov CRF (SemiCRF) [13] into an HSCRF. SemiCRF is an interesting flat segmental undirected model that generalises the chain CRF. In the SemiCRF framework the Markov process operates at the segment level, where a segment is a non-Markovian chain of nodes. A chain of segments is a Markov chain. However, since each segment can potentially have arbitrary length, inference in SemiCRFs is more involved than the chain CRFs.

Represented in our HSCRF framework (Fig. C.13), each node x_t of the SemiCRF is associated with an ending indicator e_t, with the following contextual cliques

- *Segmental state*, which corresponds to a single segment $s_{i:j}$ and is essentially the *state persistence* contextual clique in the context $c = \left[e_{i-1:j} = (1, 0, .., 0, 1)\right]$ in the HSCRF's terminology.
- *State transition*, which is similar to the state transition contextual clique in the HSCRFs, corresponding to the context $c = [e_t = 1]$.

Associated with the segmental state clique is the potential $R_{i:j}^s$, and with the state transition is the potential $A_{s',s,t}$, where $s, s' \in S$, and $S = \{1, 2, ..., K\}$.

A SemiCRF is a three-level HSCRF, where the root and bottom are dummy states. This gives a simplified way to compute the partition function, ESS, and the MAP assignment using the AIO algorithms. Thus, techniques developed in this paper for numerical scaling and partially observed data can be applied to the SemiCRF. To be more consistent with the literature of flat models such as HMMs and CRFs, we call the asymmetric inside/outside masses by the *forward/backward*, respectively. Since the model is flat, we do not need the inside and outside variables.

Forward

With some abuse of notation, let $\zeta^s_{1:j} = \left(x_{1:j-1}, e_{1:j-1}, x_j = s, e_j = 1\right)$. In other words, there is a segment of state s ending at j. We write the forward $\alpha_t(s)$ as

$$\alpha_j(s) = \sum_{\zeta^s_{1:j}} \Phi\left[\zeta^s_{1:j}, z\right] \tag{C.1}$$

As a result the partition function can be written in term of the forward as

$$\begin{aligned} Z(z) &= \sum_{\zeta_{1:T}} \Phi\left[\zeta_{1:T}, z\right] = \sum_s \sum_{\zeta^s_{1:T}} \Phi\left[\zeta^s_{1:T}, z\right] \\ &= \sum_s \alpha_T(s) \end{aligned} \tag{C.2}$$

We now derive a recursive relation for the forward. Assume that the segment ending at j starts somewhere at $i \in [1, j]$. Then for $i > 1$, there exists the decomposition $\zeta^s_{1:j} = \left(\zeta^{s'}_{1:i-1}, x_{i:j} = s, e_{i:j-1} = 0\right)$ for some s', which leads to the following factorisation

$$\Phi\left[\zeta^s_{1:j}, z\right] = \Phi\left[\zeta^{s'}_{1:i-1}\right] A_{s',s,i-1} R^s_{i:j} \tag{C.3}$$

The transition potential $A_{s',s,i-1}$ occurs in the context $c = [e_{i-1} = 1]$, and the segmental potential $R^s_{i:j}$ in the context $c = \left[x_{i:j} = s, e_{i-1} = 1, e_{i:j-1} = 0\right]$.

For $i = 1$, the factorisation reduces to $\Phi\left[\zeta^s_{1:j}, z\right] = R^s_{1:j}$. Since we do not know the starting i, we must consider all possible values in the interval $[1, j]$. Thus, Eq. (C.1) can be rewritten as

$$\alpha_j(s) = \sum_{i \in [2,j]} \sum_{s'} \sum_{\zeta^{s'}_{1:i-1}} \Phi\left[\zeta^{s'}_{1:i-1}\right] A_{s',s,i-1} R^s_{i:j} + R^s_{1:j} \tag{C.4}$$

$$= \sum_{i \in [2,j]} \sum_{s'} \alpha_{i-1}(s') A_{s',s,i-1} R^s_{i:j} + R^s_{1:j} \tag{C.5}$$

Backward

The backward is the 'mirrored' version of the forward. In particular, let $\underline{\zeta}^s_{j:T} = \left(x_{j+1:T}, e_{j:T}, x_j = s, e_{j-1} = 1\right)$. and we define the backward $\beta_t(s)$ as

$$\beta_j(s) = \sum_{\underline{\zeta}^s_{j:T}} \Phi\left[\underline{\zeta}^s_{j:T}, z\right] \tag{C.6}$$

Clearly, the partition function can be written in term of the backward as

$$Z(z) = \sum_s \beta_1(s) \tag{C.7}$$

The recursive relation for the backward

$$\beta_i(s) = \sum_{j \in [i,T-1]} \sum_{s'} R^s_{i:j} A_{s,s',j} \beta_{j+1}(s') + R^s_{i:T} \tag{C.8}$$

Typically, we want to limit the segment to the maximum length of $L \in [1, T]$. This limitation introduces some special cases when performing recursive computation of the forward and backward. Eqs. (C.4) and (C.8) are rewritten as follows

$$\alpha_j(s) = \sum_{i \in [j-L+1, j], i>1} \sum_{s'} \alpha_{i-1}(s') A_{s',s,i-1} R^s_{i:j} + R^s_{1:j} \tag{C.9}$$

$$\beta_i(s) = \sum_{j \in [i, i+L-1], j<T} \sum_{s'} R^s_{i:j} A_{s,s',j} \beta_{j+1}(s') + R^s_{i:T} \tag{C.10}$$

Finally, we can extend the HSCRF straightforwardly by allowing the bottom level states to persist. With this relaxation we have a *nested semi-Markov CRF model* in the sense that each segment in a Markov chain is also a Markov chain of sub-segments.

References

[1] Y. Bengio, learning deep architectures for AI, Found. Trends Mach. Learn. 2 (1) (2009) 1–127.
[2] S. Fine, Y. Singer, N. Tishby, The hierarchical hidden Markov model: analysis and applications, Mach. Learn. 32 (1) (1998) 41–62.
[3] G. Hinton, R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313 (5786) (2006) 504–507.
[4] R. Salakhutdinov, G. Hinton, Deep Boltzmann machines, in: Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics, AISTATS'09, vol. 5, 2009, pp. 448–455.
[5] E.F.T.K. Sang, S. Buchholz, Introduction to the CoNLL-2000 shared task: chunking, in: Proceedings of the 2nd Workshop on Learning Language in Logic and the 4th Conference on Computational Natural Language Learning, Lisbon, Portugal, vol. 7, 2000, pp. 127–132, http://www.cnts.ua.ac.be/conll2000/chunking/.
[6] N. Oliver, A. Garg, E. Horvitz, Layered representations for learning and inferring office activity from multiple sensory channels, Comput. Vis. Image Underst. 96 (2) (2004) 163–180.
[7] J. Lafferty, A. McCallum, F. Pereira, Conditional random fields: probabilistic models for segmenting and labeling sequence data, in: Proceedings of the International Conference on Machine Learning, ICML, 2001, pp. 282–289.
[8] C. Sutton, A. McCallum, K. Rohanimanesh, Dynamic conditional random fields: factorized probabilistic models for labeling and segmenting sequence data, J. Mach. Learn. Res. 8 (Mar 2007) 693–723.
[9] L. Liao, D. Fox, H. Kautz, Extracting places and activities from GPS traces using hierarchical conditional random fields, Int. J. Robot. Res. 26 (Jan 2007) 119–134.
[10] S. Kumar, M. Hebert, A hierarchical field framework for unified context-based classification, in: Proceedings of the IEEE International Conference on Computer Vision, ICCV, vol. 2, Oct 2005, pp. 1284–1291.
[11] D. Yu, S. Wang, L. Deng, Sequential labeling using deep-structured conditional random fields, IEEE J. Sel. Top. Signal Process. 4 (6) (2010) 965–973.
[12] T. Truyen, D. Phung, H. Bui, S. Venkatesh, Hierarchical semi-Markov conditional random fields for recursive sequential data, in: Twenty-Second Annual Conference on Neural Information Processing Systems, NIPS, Vancouver, Canada, Dec 2008, pp. 1657–1664.
[13] S. Sarawagi, W.W. Cohen, Semi-Markov conditional random fields for information extraction, in: L.K. Saul, Y. Weiss, L. Botton (Eds.), Advances in Neural Information Processing Systems 17, MIT Press, Cambridge, Massachusetts, 2004, pp. 1185–1192.
[14] S. Lauritzen, Graphical Models, Oxford Science Publications, 1996.
[15] H.H. Bui, D.Q. Phung, S. Venkatesh, Hierarchical hidden Markov models with general state hierarchy, in: D.L. McGuinness, G. Ferguson (Eds.), Proceedings of the 19th National Conference on Artificial Intelligence, AAAI, San Jose, CA, Jul 2004, pp. 324–329.
[16] L.R. Rabiner, A tutorial on hidden Markov models and selected applications in speech recognition, Proc. IEEE 77 (2) (1989) 257–286.
[17] K. Murphy, M. Paskin, Linear Time Inference in Hierarchical HMMs, Advances in Neural Information Processing Systems (NIPS), vol. 2, MIT Press, 2002, pp. 833–840.
[18] P.Q. Dinh, Probabilistic and Film Grammar Based Methods for Video Content Understanding, PhD thesis, Curtin University of Technology, 2005.
[19] B. Taskar, P. Abbeel, D. Koller, Discriminative probabilistic models for relational data, in: Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence, UAI-02, Morgan Kaufmann, 2002, pp. 485–492.
[20] J. Pearl, Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, Morgan Kaufmann, San Francisco, CA, 1988.
[21] N. Nguyen, D. Phung, S. Venkatesh, H.H. Bui, Learning and detecting activities from movement trajectories using the hierarchical hidden Markov models, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR, San Diego, CA, vol. 2, Jun 2005, pp. 955–960.
[22] T. Truyen, D. Phung, H. Bui, S. Venkatesh, AdaBoost.MRF: boosted Markov random forests and application to multilevel activity recognition, in: Computer Vision and Pattern Recognition, New York, USA, vol. 2, June 2006, pp. 1686–1693.
[23] E. Brill, Transformation-based error-driven learning and natural language processing: a case study in part-of-speech tagging, Comput. Linguist. 21 (4) (1995) 543–566.
[24] S.V.N. Vishwanathan, N.N. Schraudolph, M.W. Schmidt, K.P. Murphy, Accelerated training of conditional random fields with stochastic gradient methods, in: Proceedings of the International Conference on Machine Learning, ICML, 2006, pp. 969–976.
[25] H.H. Bui, S. Venkatesh, G. West, Policy recognition in the abstract hidden Markov model, J. Artif. Intell. Res. 17 (2002) 451–499.
[26] K. Murphy, Dynamic Bayesian Networks: Representation, Inference and Learning, PhD thesis, Computer Science Division, University of California, Berkeley, Jul 2002.
[27] F. Pereira, Y. Schabes, Inside–outside reestimation from partially bracketed corpora, in: Proceedings of the Meeting of the Association for Computational Linguistics, ACL, 1992, pp. 128–135.
[28] J. Chung, S. Ahn, Y. Bengio, Hierarchical multiscale recurrent neural networks, arXiv preprint, arXiv:1609.01704, 2016.
[29] S. El Hihi, Y. Bengio, Hierarchical recurrent neural networks for long-term dependencies, in: Advances in Neural Information Processing Systems 8, NIPS'95, MIT Press, Cambridge, MA, 1996, pp. 493–499.
[30] N.V. Cuong, N. Ye, W.S. Lee, H.L. Chieu, Conditional random field with high-order dependencies for sequence labeling and segmentation, J. Mach. Learn. Res. 15 (2014) 981–1009.
[31] X. Qian, X. Jiang, Q. Zhang, X. Huang, L. Wu, Sparse higher order conditional random fields for improved sequence labeling, in: Proceedings of the 26th Annual International Conference on Machine Learning, ACM, 2009, pp. 849–856.
[32] Y. Miyao, J. Tsujii, Maximum entropy estimation for feature forests, in: Proceedings of Human Language Technology Conference, HLT, Morgan Kaufmann Publishers Inc., 2002, pp. 292–297.
[33] S. Clark, J.R. Curran, Log-linear models for wide-coverage CCG parsing, in: Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP, 2003, pp. 97–104.
[34] J. Yedidia, W. Freeman, Y. Weiss, Constructing free-energy approximations and generalized belief propagation algorithms, IEEE Trans. Inf. Theory 51 (7) (2005) 2282–2312.
[35] F.R. Kschischang, B.J. Frey, H.A. Loeliger, Factor graphs and the sum-product algorithm, IEEE Trans. Inf. Theory 47 (February 2001) 498–519.
[36] T.T. Truyen, On Conditional Random Fields: Applications, Feature Selection, Parameter Estimation and Hierarchical Modelling, PhD thesis, Curtin University of Technology, 2008.
[37] T. Truyen, D. Phung, S. Venkatesh, H. Bui, MCMC for hierarchical semi-Markov conditional random fields, in: NIPS'09 Workshop on Deep Learning for Speech Recognition and Related Applications, Whistler, BC, Canada, Dec 2009.