Entropy Semiring Forward-backward Algorithm for HMM Entropy Computation

Velimir M. Ilić

Abstract—The paper presents Entropy Semiring Forward-backward algorithm (ESRFB) and its application for memory efficient computation of the subsequence constrained entropy and state sequence entropy of a Hidden Markov Model (HMM) when an observation sequence is given. ESRFB is based on forward-backward recursion over the entropy semiring, having the lower memory requirement than the algorithm developed by Mann and MacCallum, with the same time complexity. Furthermore, when it is used with forward pass only, it is applicable for the computation of HMM entropy for a given observation sequence, with the same time and memory complexity as the previously developed algorithm by Hernando et al.

I. INTRODUCTION

Hidden Markov Models (HMMs) are standard probabilistic models for state sequences in sequential data labeling [10]. Subsequence constrained entropy of HMM explaining an observation sequence and state sequence entropy, are useful quantities which provide a measure of HMM uncertainty. One criterion for the estimation of the HMM quality is the entropy of state sequence explaining an observation sequence, which provides a measure of its uncertainty [2], [6].

The algorithms for HMMs mostly consider efficient marginalization which is usually performed using the forward-backward algorithm ([3]), which runs in $O(N^2T)$ time, where $N$ denotes the number of states and $T$ is the length of sequence. Recently, Mann and MacCallum have developed an algorithm for computation of HMM subsequence constrained entropy for similar probabilistic model conditional random fields (CRF), which is based on the marginal probabilities computation [9] with the same asymptotical complexity as $FB$. This algorithm can be adapted to work with HMMs, but when the sequence length is large it becomes memory demanding, since it needs $O(NT)$ memory. On the other hand, Hernando et al. [6] developed the memory efficient algorithm for state sequence entropy computation which requires $O(N)$ memory. The algorithm has the same time complexity as $FB$, but it is not applicable for the computation of subsequence constrained entropy.

In this paper we develop a new algorithm which can be used for both types of computations. The algorithm is based on forward-backward recursion over the entropy semiring [7] and is called Entropy Semiring Forward-backward algorithm (ESRFB). ESRFB has lower memory requirement than Mann-MacCallum’s algorithm subsequence constrained entropy computation. Furthermore, when it is used with the forward pass only it can compute the entropy in the same time and space as Hernando et al.’s algorithm. Moreover, it is shown how the Hernando et al.’s algorithm can be derived from ESRFB.

The paper is organized as follows. In section II we define the HMM and present the forward-backward algorithm (FB) for efficient marginalization of HMM. Section III reviews the algorithms by Hernando et. al. and Mann and McCullam, for efficient computation of HMM entropy and subsequence constrained entropy for a given observation sequence. Section IV gives the general FB algorithm which operates over the commutative semiring. Finally, section V considers the FB over the entropy semiring and its application to HMM entropy computation.

II. HIDDEN MARKOV MODELS AND FORWARD-BACKWARD ALGORITHM

In this paper, we adopt the following notation:

- The sequence $l, l+1, \ldots, r$ is shortly denoted with $l : r$, and the sequence $0 : l, r : T$ is denoted with $-l : r$.
- Big letters are used for random variables $(S_t, O_t)$ and the small ones for their realizations $(s_t, o_t)$.
- The sequence of symbols is $(S_1, \ldots, S_r)$ is denoted with $S_{l:r}$, the sequence $(S_0, \ldots, S_r, S_{r+1}, \ldots, S_T)$ with $S_{l:T}$ and similarly for $O_{l:r}$ and $o_{l:r}$ for $0 \leq l \leq r \leq T$.
- The sequences $S_{0:T}, O_{0:T}, s_{0:T}$ and $o_{0:T}$ are denoted with $S, O, s$ and $o$, respectively.
- The variables are omitted in probability notation. Thus, $p(s_{t}, o_{1:t})$ stands for $p(S_t = s_t, O_{1:t} = o_{1:t})$, $p(o)$ for $p(O = o)$ and so on.

Hidden Markov model (HMM) consists of the following elements:

- A Markov chain $(S_0, \ldots, S_T)$, represented by an $N \times N$ stochastic matrix $A$, which describes the transition probabilities $a_{ij} = P(S_t = j | S_{t-1} = i)$ between the $N$ states of the model, together with a probability distribution $b_i$, where $\pi_t = p(S_0 = i)$.
- A set of probability distributions, one for each hidden state, $b_i(o_t) = P(O_t = o_t | S_t = i)$, which model the emission of such observations. If there are $M$ possible distinct observations, we accommodate the probability distributions to be in the rows of an $N \times M$ matrix $B$.

With these settings, the joint probability that state sequence $S$ takes value $s$ and the observation sequence $O$ takes value $o$ is given with:

$$p(s, o) = \pi_{s_0} b_{s_0}(o_t) \prod_{t=1}^{T} a_{s_{t-1:s_t}}(o_t).$$

Using the probability conditions $\sum_{s_t} a_{s_{t-1}, s_t} = 1$ and $\sum_{o_t} b_{s_t}(o_t) = 1$, we can derive two important equations which
characterizes HMM and will be used in the rest of the paper:

\[
p(s_{0:i}, o_{0:i}) = \pi_{s_0} b_{s_0}(o_0) \prod_{t=1}^{i} a_{s_{t-1:s_t}} b_{s_t}(o_t),
\]

\[
p(s_{i+1:T}, o_{i+1:T}|s_t) = \prod_{t=i+1}^{T} a_{s_{t-1:s_t}} b_{s_t}(o_t).
\]

One of the main problems in HMMs is efficient marginalization of computation of the HMM conditional probability \(p(s|o)\):

\[
p(s_{t:r}|o) = \sum_{s_{t-1}} p(s|o) = \sum_{s_{t-1}} \frac{p(s,o)}{p(o)}.
\]

The computation of (4) by enumerating all the \(s \in S^{T+1}\) requires about \(T N^{T+1}\) additions and multiplications, which would be infeasible even for small values of \(N\) and \(T\) (for \(N = 10\) and \(T = 20\), the total number of operations has an order \(10^{22}\)). A more efficient way is the forward-backward (FB) algorithm which solves the problems by use of \(O(N^2 T)\) operations. In this paper we present a numerical stable variant of FB. For another variants see \([10, 2, 3]\)

The forward-backward algorithm recursively computes desired quantities using the HMM forward and backward probabilities:

\[
\hat{\alpha}_t(s_t) = p(s_t|o_{1:t}),
\]

\[
\hat{\beta}_t(s_t) = \frac{p(o_{t+1:T}|s_t)}{p(o_{t+1:T}|o_{0:t})}.
\]

as follows.

1) Forward initialization: For \(1 \leq j \leq N\):

\[
c_0 = \sum_{j=1}^{N} \pi_j b_j(o_0), \quad \hat{\alpha}_0(j) = \frac{\pi_j b_j(o_0)}{c_0}.
\]

2) Forward recursion: For \(0 \leq t \leq T, 1 \leq j \leq N\):

\[
c_t = \sum_{j=1}^{N} \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_j(o_t),
\]

\[
\hat{\alpha}_t(j) = \frac{\sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_j(o_t)}{c_t}.
\]

3) Backward initialization: For \(1 \leq i \leq N\):

\[
\hat{\beta}_T(i) = 1,
\]

4) Backward recursion: For \(T - 1 \geq t \geq 0, 1 \leq i, j \leq N\):

\[
\hat{\beta}_t(i) = \frac{\sum_{j=1}^{N} a_{ij} b_j(o_{t+1}) \hat{\beta}_{t+1}(j)}{c_{t+1}}.
\]

The normalization factors \(c_t\) ensure that the probabilities sums to one and represents the conditional observational probabilities:

\[
c_0 = p(o_0), \quad c_t = p(o_t|o_{0:t-1}).
\]

Once the forward and backward probabilities are computed we can compute the marginal as

\[
p(s_{t:r}|o) = \alpha_t(s_t) \cdot \prod_{t=l+1}^{r} a_{s_{t-1:s_t}} b_{s_t}(o_t) \cdot \beta_t(s_r).
\]

Two most commonly used marginals \(p(s_{t-1:t}|o)\) and \(p(s_t|o)\) can be computed as follows

\[
p(s_{t-1:t}|o) = \hat{\alpha}_{t-1}(s_{t-1})a_{s_{t-1:s_t}} b_{s_t}(o_t) \hat{\beta}_t(s_t),
\]

\[
p(s_t|o) = \hat{\alpha}_t(s_t) \cdot \hat{\beta}_t(s_t).
\]

The majority of computations are performed in the forward and backward recursion phases, which results in the time complexity \(O(N^2 T)\). The storing of all forward and backward vectors along with the normalization factors requires \(O(NT)\) memory.

### III. Entropy computation of Hidden Markov Models

The conditional entropy of HMM is given with

\[
H(S \mid o) = - \sum_s p(s \mid o) \log p(s \mid o).
\]

while, the subsequence constrained entropy is

\[
H(S_{t:r} \mid s_{t:r}, o_0:T) = - \sum_{s_{t:r}} p(s_{t:r} \mid s_{t:r}, o_0:T) \cdot \log p(s_{t:r} \mid s_{t:r}, o_0:T).
\]

If we introduce

\[
H(S_{t:r}, s_{t:r} \mid o) = - \sum_{s_{t:r}} p(s \mid o) \cdot \log p(s \mid o),
\]

we can derive the following equality

\[
H(S_{t:r} \mid s_{t:r}, o) = \frac{H(S_{t:r} \mid o) + \log p(s_{t:r} \mid o)}{p(s_{t:r} \mid o)}.
\]

A direct evaluation of (15) is infeasible as there are \(N^T\) terms. In the following text we consider efficient algorithms for the entropy computation.

First, in the next two subsections, we review two algorithms based on the entropy decomposition rules \([4]\)

\[
H(X, Y) = H(X) + H(Y \mid X),
\]

\[
H(Y \mid X) = \sum_x p(x) \cdot H(Y \mid X = x).
\]

After that, in the next section we derive the new algorithms based on the ESRFB algorithm.

#### A. The algorithm by Mann and McCallum

Mann and McCallum proposed the algorithm for the linear chain conditional random fields entropy gradient computation \([9]\), which can also be used for HMMs. The algorithm uses the conditional probabilities

\[
\hat{p}_{l[t+1]}(i|j) = \frac{p(s_t|s_{t+1}, o)}{p(s_t|o)},
\]

\[
\hat{p}_{l[t]}(i|j) = \frac{p(s_{t-1:t}|o)}{p(s_{t-1}|o)},
\]

which, in turn, are computed using the FB algorithm and the forward and backward entropies, which, in turn, are computed with the recursive procedure based on the entropy.
decomposition formulas \[19\]. The forward entropy \(H^t(s_t)\) at time \(t\) is defined as the entropy of state sequence \(S_{0:t-1}\) which ends in \(s_t\), for a given observation sequence \(o\):

\[
H^t_o(s_t) = H(S_{0:t-1}|s_t, o),
\]

while the backward entropy \(H^B_{t} (s_t)\) at time \(t\) is the entropy of state sequence \(S_{t+1:T}\) which starts in \(s_t\):

\[
H^B_{t}(s_t) = H(S_{t+1:T}|s_t, o).
\]

Using the forward and backward entropies, subsequence constrained entropy conditional HMM entropy can be recursively computed as in the following algorithm.

1) Forward backward algorithm: Compute and store forward and backward probabilities using \(FB\) algorithm.

2) Forward entropy initialization: For \(1 \leq j \leq N\):

\[
H^F_0(j) = 0;
\]

3) Forward entropy recursion: for \(0 \leq t \leq T-1, 1 \leq i, j \leq N\):

\[
H^F_{t+1}(j) = \sum_{i=1}^{N} p_{t+1|i}(i|j) \left( H^F_{t}(i) - \log p_{t+1|i}(i|j) \right).
\]

4) Backward entropy initialization: For \(1 \leq j \leq N\):

\[
H^B_{1}(j) = 0;
\]

5) Backward entropy recursion: for \(0 \leq t \leq T-1, 1 \leq i, j \leq N\):

\[
H^B_{t-1}(j) = \sum_{i=1}^{N} p_{t-1|i}(i|j) \left( H^B_{t}(i) - \log p_{t-1|i}(i|j) \right).
\]

The time complexity of the algorithm is \(O(N^2 T + N^{T-1})\), where \(O(N^2 T)\) is for the forward-backward entropy computation and \(O(N^{T-1})\) for the termination phase. The memory complexity depends on the sequence length since all forward and backward vectors should be available in forward and backward recursion phases; regarding \(O(N^{T-1})\) space required for storing the results in the termination phase, the total memory complexity is \(O(NT + N^{T-1})\).

The algorithm can also be used for the computation of entropy using the equality

\[
H(S|o) = H(S_T|o) + \sum_{s_{0:T-1}} p(s_{0:T}|o) \cdot H^F_T(s_T),
\]

which follows from the entropy decomposition formulas and definition of forward entropy. In this case, the backward entropy pass is not needed, but the time and memory complexity are not reduced, since the forward and backward probabilities still need to be computed. In the following subsection we review the algorithm developed in \[6\] by Hernando et al., which computes the entropy with the memory complexity independent of the sequence length.

B. The algorithm by Hernando et al.

In \[6\], Hernando et al. develop the recursive algorithm for the computation of Hidden Markov model entropy. It uses HMM forward probability

\[
\hat{\alpha}_t(s_t) = p(s_t|o_{1:t}),
\]

conditional probability

\[
\hat{p}_{t|i-1}(s_t|s_{i-1}) = p(s_t|s_{i-1}, o_{1:t}),
\]

and intermediate entropy

\[
H^F_t(s_t) = H(S_{0:t-1}|s_t, o_{1:t}).
\]

HMM entropy is computed as follows.

1) Initialization: For \(1 \leq j \leq N\) set:

\[
H^F_0(j) = 0,
\]

\[
\hat{\alpha}_0(j) = \frac{\pi_j b_j(o_0)}{\sum_{i=1}^{N} \pi_i b_i(o_1)}.
\]

2) Induction: For \(1 \leq t \leq T\) and \(1 \leq j \leq N\) set:

\[
\hat{\alpha}_t(j) = \frac{\sum_{i=1}^{N} \hat{\alpha}_{t-1}(i) a_{ij} b_j(o_t)}{\sum_{k=1}^{N} \sum_{i=1}^{N} \hat{\alpha}_{t-1}(i)a_{ik} b_k(o_t)},
\]

\[
\hat{p}_{t-1|i}(i|j) = \frac{\hat{\alpha}_{t-1}(i) a_{ij}}{\sum_{k=1}^{N} \sum_{i=1}^{N} \hat{\alpha}_{t-1}(i) a_{ik} k},
\]

\[
H_t(j) = \sum_{i=1}^{N} \hat{p}_{t-1|i}(i|j) \left( H_{t-1}(i) - \log \hat{p}_{t-1|i}(i|j) \right).
\]

3) Termination:

\[
H(S|o) = \sum_{j=1}^{N} \hat{\alpha}_T(j) \left( H_T(j) - \log \hat{\alpha}_T(j) \right).
\]

The algorithm runs with the linear time complexity \(O(N^2 T)\) and fixed memory space independent of sequence length, \(O(N^2)\), since the vectors \(\hat{\alpha}_{t-1}, H_{t-1}\) and the matrix \(p_{t-1|i}\) should be computed only once in \(t-1\)-th iteration and, after having been used for the computation of \(H_t\), they can be deleted.

IV. THE FORWARD-BACKWARD OVER THE COMMUTATIVE SEMiring

The FB algorithm for HMMs works for more general models in which the factors in \(1\) are not probabilities but the functions whose range is a commutative semiring \(1\). In this section we present the forward-backward over the commutative semiring and derive the \(FB\) for HMMs as a special case.
A. The forward-backward algorithm over a commutative semiring

We begin with the definition of the commutative semiring.

**Definition 1.** A commutative semiring is a set $\mathbb{K}$ with operations $\oplus$ and $\otimes$ such that both $\oplus$ and $\otimes$ are commutative and associative and have identity elements in $\mathbb{K}$ ($\mathbb{0}$ and $\mathbb{T}$ respectively), and $\otimes$ is distributive over $\oplus$.

Let $s = \{s_0, \ldots, s_T\}$ be a set of variables taking values from the set $S = \{1, \ldots, N\}$. We define the local kernel functions, $u_0 : S \to \mathbb{K}$, $u_t : S^2 \to \mathbb{K}$ for $t = 1, \ldots, T$, and the global kernel function $u : S^{T+1} \to \mathbb{K}$, assuming that the following factorization holds

$$u(s) = u_0(s_0) \otimes \prod_{t=1}^{T} u_t(s_{t-1}, s_t)$$

for all $s = (s_0, \ldots, s_T) \in S^{T+1}$.

The FB algorithm solves two problems

1) The marginalization problem: Compute the sum

$$v_{a:b}(s_{a:b}) = \bigoplus_{s_{0:T} \to a:b} u(s),$$

2) The normalization problem: Compute the sum

$$Z = \bigoplus_s u(s).$$

Similarly as in HMM, the FB recursively computes the forward variable

$$\alpha_t(s_t) = \bigoplus_{s_{0:t-1}} u_0(s_0) \otimes \prod_{t=1}^{i} u_t(s_{t-1}, s_t)$$

which is initialized to

$$\alpha_0(s_0) = u_0(s_0),$$

and recursively computed using

$$\alpha_t(s_t) = \bigoplus_{s_{t+1:T}} \otimes u_t(s_{t-1}, s_t)$$

and the backward variable

$$\beta_t(s_t) = \bigoplus_{s_t+1} \otimes u_t(s_{t-1}, s_t),$$

which is recursively computed using

$$\beta_t(s_t) = \bigoplus_{s_{t+1:T}} \otimes u_t(s_{t-1}, s_t) \otimes \beta_{t+1}(s_{t+1}),$$

and initialized to

$$\beta_T(s_T) = 1.$$

Once, the forward $\alpha_t$ and backward $\beta_t$ variables are computed, we can solve the marginalization problem by use of the formula

$$v_{1:T}(s_{1:T}) = \alpha_T(s_T) \otimes \prod_{t=t+1}^{r} u_t(s_{t-1}, s_t) \otimes \beta_T(s_T).$$

The normalization problem can be solved with the forward pass only according to

$$\bigoplus_s u(s) = \bigoplus_{s_{1:T}} \alpha_T(s_T).$$

In the following subsection we derive the FB algorithm for HMMs as a special case of the FB over the commutative semiring.

B. HMM forward-backward as a special case of the forward-backward over the commutative semiring

The conditional HMM probability $p(s,o)$ can be seen as a special case of the global kernel factorization (2) if $\oplus$ and $\otimes$ stand for the addition and multiplication of the real numbers. To clarify this, recall that HMM probability (1) has the form

$$p(s,o) = \pi_s b_s(o_0) \prod_{t=1}^{T} a_{s_{t-1}, s_t} b_{s_t}(o_t),$$

and that according to the chain rule, conditional observational probability can be represented as

$$p(o_{0:T}) = p(o_0) \cdot \prod_{t=1}^{T} p(o_t|o_{0:t-1}) = c_0 \cdot \prod_{t=1}^{T} c_t,$$

where $c_0 = p(o_0)$ and $c_t = p(o_t|o_{0:t-1})$ as in (11). Then,

$$p(s|o) = \frac{p(s,o)}{p(o)} = z_0(s_0) \prod_{t=1}^{T} z_t(s_{t-1}, s_t),$$

where

$$z_0(s_0) = \frac{\pi_s b_s(o_0)}{c_0}, \quad z_t(s_{t-1}, s_t) = \frac{a_{s_{t-1}, s_t} b_{s_t}(o_t)}{c_t}.$$

According to the equation (2), the subsequence conditional probabilities can be represented as $p(s_{0:t}|o_{0:t-1}) = z_0(s_0) \prod_{i=1}^{t} z_i(s_{i-1}, s_i)$, and the forward variable has the form

$$\alpha_t(s_t) = \sum_{s_{0:t-1}} z_0(s_0) \prod_{i=t}^{T} z_i(s_{i-1}, s_t) = p(s_t|o_{0:t}),$$

in agreement with (5). The recursive equations (46), (45) for the forward variable have the form

$$\alpha_0(s_0) = z_0(s_0) = \frac{\pi_s b_s(o_0)}{c_0},$$

$$\alpha_t(s_t) = \sum_{s_{t+1:T}} \frac{z_t(s_{t-1}, s_t) \cdot \alpha_{t+1}(s_{t+1})}{c_t} = \frac{\sum_{s_{t+1}} a_{s_{t-1}, s_t} b_{s_t}(o_t) \alpha_{t+1}(s_{t+1})}{c_t},$$

and the normalization factors can be computed using the probability condition

$$\sum_{s_t} \alpha_t(s_t) = \sum_{s_t} p(s_t|o_{0:t}) = 1,$$

which gives

$$c_0 = \sum_{j=1}^{N} \pi_j b_j(o_0), \quad c_t = \sum_{j=1}^{N} \sum_{i=1}^{N} \alpha_{t+1}^{(z)}(i) a_{ij} b_j(o_t).$$
and the definition of forward variable and its recursive equations agrees with the equations from section III. Similarly, according to the equation (63), \( p(s_{i+1:T}, o_{i+1:T} | s_i) = \prod_{t=i+1}^{T} z_t(s_{t-1}, s_t) \), the backward variable is

\[
\beta_t(s_i) = \sum_{s_{t+1:T}} \prod_{t=i+1}^{T} z_t(s_{t-1}, s_t) \frac{p(o_{t+1:T} | s_t)}{p(o_{i+1:T} | o_{0:t})}.
\]

(61)

with the recursive equation:

\[
\beta_T(s_T) = 1,
\]

\[
\beta_t(s_i) = \frac{\sum_{s_{t+1}} a_{s_{t+1}, s_i} b_{n_t}(o_t) \beta_{t+1}(s_{t+1})}{c_t}.
\]

(62)

Finally, in the same manner, the equation (60) reduces to (64)

\[
p(s_{i:t} | o) = \alpha_t(s_t) \prod_{t=1+1}^{r} a_{s_{t-1}, s_t} b_{n_t}(o_t) \beta_t(s_{t+1})
\]

(63)

which retrieves the HMM forward-backward algorithm.

V. THE FORWARD-BACKWARD ALGORITHM OVER THE ENTROPY SEMIRING

In this section we consider the forward-backward algorithm over the entropy semiring (ESRFB) and its application to HMM entropy computation. The entropy semiring (ESR), which is introduced in [3] and [5], is defined as follows.

**Definition 2:** The entropy semiring is a the commutative semiring for which \( K = \mathbb{R}^2 \) and the semiring operations are defined with:

\[
(z_1, h_1) \oplus (z_2, h_2) = (z_1 + z_2, h_1 + h_2),
\]

\[
(z_1, h_1) \oplus (z_2, h_2) = (z_1 z_2, z_1 h_2 + z_2 h_1),
\]

(65)

(66)

for all \((z_1, h_1), (z_2, h_2) \in \mathbb{R}^2\). The identities for \( \oplus \) and \( \otimes \) are \((0,0)\) and \((1,0)\), respectively.

The first component of an ordered pair is called a \( z \)-part, while the second one is an \( h \)-part. The following lemma can be proven by the induction (see [7]).

**Lemma 1:** Let \((z_i, h_i) \in \mathbb{R}^2\) for all \(0 \leq i \leq T\). Then, the following equality holds:

\[
\bigotimes_{i=0}^{T} (z_i, h_i) = \left( \prod_{i=0}^{T} z_i, \prod_{i=0}^{T} z_i \cdot \prod_{j=0}^{T} h_j \right).
\]

(67)

Let the local kernels in (41) have the form:

\[
u_0(s_0) = (z_0(s_0), z_0(s_0) h_0(s_0)),
\]

\[
u_t(s_{i-1}, s_i) = (z_t(s_{i-1}, s_i), z_t(s_{i-1}, s_i) h_t(s_{i-1}, s_i)),
\]

(68)

(69)

where

\[
z_0(s_0) = \frac{\pi_{s_0} b_{n_0}(o_0)}{c_0}, \quad z_t(s_{t-1}, s_t) = \frac{a_{s_{t-1}, s_t} b_{n_t}(o_t)}{c_t}.
\]

(70)

with \( c_0 = p(o_0) \), \( c_t = p(o_t | o_{0:t-1}) \) and

\[
h_0(s_0) = \log z_0(s_0), \quad h_t(s_{t-1}, s_t) = \log z_t(s_{t-1}, s_t).
\]

(71)

From Lemma 1, it follows that the \( z \) and \( h \) parts of the global kernel

\[
u(s) = \nu_0(s_0) \bigotimes_{i=1}^{T} \nu_t(s_{i-1}, s_i),
\]

(72)

are given with:

\[
u(s)^{(z)} = z_0(s_0) \prod_{t=1}^{T} z_t(s_{i-1}, s_i)
\]

(73)

\[
u(s)^{(h)} = z_0(s_0) \prod_{t=1}^{T} z_t(s_{i-1}, s_i)
\]

(74)

\[
h_0(s_0) + \sum_{j=1}^{T} h_j(s_{j-1}, s_j) = \log \left( z_0 \cdot \prod_{j=1}^{T} z_j(s_{j-1}, s_j) \right),
\]

(75)

Note that

\[
h_0(s_0) + \sum_{j=1}^{T} h_j(s_{j-1}, s_j) = \log \left( z_0 \cdot \prod_{j=1}^{T} z_j(s_{j-1}, s_j) \right),
\]

(76)

and, according to the factorization (64) for HMM conditional probability \( p(s | o) = z_0 \cdot \prod_{t=1}^{T} z_t(s_{i-1}, s_i) \), we can represent the global kernel as follows

\[
u(s) = \left( p(s | o), p(s | o) \log p(s | o) \right).
\]

(77)

Hence, by summing of the global kernel we can obtain the entropies \( H(S | o) \) or \( H(S_{t-1:t}, r_{t-1:t} | o) \) as the \( h \) part of the sum, which depends on the set of the variables which are summed out. Two types of the summation correspond to the normalization and marginalization of the global kernel which can be solved with the forward-backward algorithm over the entropy semiring.

The \( z \) and \( h \) parts of the forward and backward variables in the entropy semiring can also be derived using Lemma I.

For the forward vector,

\[
\alpha_t(s_t) = \bigoplus_{s_{0:t-1}} \nu_0(s_0) \bigotimes_{i=1}^{t} \nu_t(s_{i-1}, s_i)
\]

(78)

we have

\[
\alpha_t^{(z)}(s_t) = \sum_{s_{0:t-1}} \nu_0(s_0) \prod_{i=1}^{t} \nu_t(s_{i-1}, s_i)
\]

(79)

\[
\alpha_t^{(h)}(s_t) = \sum_{s_{0:t-1}} \nu_0(s_0) \prod_{i=1}^{t} \nu_t(s_{i-1}, s_i)
\]

(80)

\[
h_0(s_0) + \sum_{j=1}^{t} h_j(s_{j-1}, s_j),
\]

(81)

and by use of the equality \( p(s_{0:t} | o_{0:t}) = z_0(s_0) \prod_{t=1}^{T} z_t(s_{i-1}, s_i) \), we obtain

\[
\alpha_t^{(z)}(s_t) = \sum_{s_{0:t}} p(s_{0:t} | o_{0:t}) = p(s_t | o_{0:t})
\]

(82)

\[
\alpha_t^{(h)}(s_t) = \sum_{s_{0:t}} p(s_{0:t} | o_{0:t}) \log p(s_{0:t} | o_{0:t})
\]

(83)

The \( z \)-part of the ESR forward vector is the HMM forward probability as defined in the sections III and V-B while the information about subsequence entropies is propagated through the \( h \)-part, so that at each step we have

\[
H(S_{0:t} | o_{0:t}) = \sum_{s_t} \alpha_t^{(h)}(s_t).
\]

(84)
The forward vector is initialized to \( u_0(s_0) \) and regarding \((85)\) we have:
\[
\alpha_0^{(z)}(s_0) = z_0(s_0) = \frac{\pi_{s_0} b_{s_0}(o_0)}{c_0},
\]
\[
\alpha_0^{(h)}(y_0) = z_0(s_0) h_0(s_0) = \frac{\pi_{s_0} b_{s_0}(o_0)}{c_0} \log \frac{\pi_{s_0} b_{s_0}(o_0)}{c_0}.
\]

The \( z \) and \( h \) forward recursive equation
\[
\alpha_i(s_i) = \bigoplus_{s_{i-1}} u_{i-1}(s_{i-1}, s_i) \otimes \alpha_{i-1}(s_{i-1}),
\]
can be determined using the definition of the entropy semiring as
\[
\alpha_i^{(z)}(s_i) = \sum_{s_{i-1}} z_i(s_{i-1}, s_i) \alpha_{i-1}^{(z)}(s_{i-1})
\]
\[
\alpha_i^{(h)}(s_i) = \sum_{s_{i-1}} z_i(s_{i-1}, s_i) \cdot (\alpha_{i-1}^{(h)}(s_{i-1}) + h_i(s_{i-1}, s_i) \alpha_{i-1}^{(z)}(s_{i-1})),
\]
or equivalently
\[
\alpha_i^{(z)}(j) = \sum_{i=1}^{N} a_{ij} b_j(o_i) \cdot \alpha_{i-1}^{(z)}(i)
\]
\[
\alpha_i^{(h)}(j) = \sum_{i=1}^{N} a_{ij} b_j(o_i) \cdot (\alpha_{i-1}^{(h)}(i) + \alpha_{i-1}^{(z)}(i) \log \frac{a_{ij} b_j(o_i)}{c_t}).
\]

Similarly as in sections [II] and [IV-B], factors \( c_t \) can be found by normalization of \( z \)-parts:
\[
c_0 = \sum_{j=1}^{N} \pi_j b_j(o_0) \quad c_t = \sum_{j=1}^{N} \sum_{i=1}^{N} \alpha_{i-1}^{(z)}(i) a_{ij} b_j(o_i).
\]

The backward vector
\[
\beta_t(s_t) = \bigoplus_{s_{t+1}} u_{t+1}(s_{t+1}, s_t),
\]
has corresponding \( z \) and \( h \) parts
\[
\beta_t^{(z)}(s_t) = \prod_{s_{t+1}}^T z_i(s_{i-1}, s_i)
\]
\[
\beta_t^{(h)}(s_t) = \prod_{s_{t+1}}^T z_i(s_{i-1}, s_i) \cdot \prod_{j=i+1}^T h_j(s_{j-1}, s_j).
\]
The equality \((83)\) implies
\[
\prod_{i=t+1}^T z_i(s_{i-1}, s_i) = \frac{p(s_{t+1:T}, o_{t+1:T}|s_t)}{p(o_{t+1:T}|o_{0:t})}
\]
and we have
\[
\beta_t^{(z)}(s_t) = \frac{p(o_{t+1:T}|s_t)}{p(o_{t+1:T}|o_{0:t})}
\]
\[
\beta_t^{(h)}(s_t) = \sum_{s_{t+1}} p(s_{t+1:T}, o_{t+1:T}|s_t) \log \frac{p(s_{t+1:T}, o_{t+1:T}|s_t)}{p(o_{t+1:T}|o_{0:t})},
\]
which gives the \( z \)-part of the ESR backward vector, the same as HMM backward probability from the sections [II] and [IV-B]. The backward vector is initialized according to \( \beta_T(s_T) = 1 \):
\[
\beta_T(s_T)^{(z)} = 1, \quad \beta_T(s_T)^{(h)} = 0.
\]
while the recursive equation
\[
\beta_t(s_t) = \bigoplus_{s_{t+1}} u_{t+1}(s_{t+1}, s_t) \otimes \beta_{t+1}(s_{t+1}),
\]
reduces to
\[
\beta_t^{(z)}(s_t) = \sum_{s_{t+1}} z(s_t, s_{t+1}) \beta_{t+1}^{(z)}(s_{t+1})
\]
\[
\beta_t^{(h)}(s_t) = \sum_{s_{t+1}} z(s_t, s_{t+1}) \cdot (\beta_{t+1}^{(h)}(s_{t+1}) + h(s_t, s_{t+1}) \alpha_{t+1}^{(z)}(s_{t+1})),
\]
or equivalently
\[
\beta_t^{(z)}(i) = \sum_{j} a_{ij} b_j(o_t) \beta_{t+1}^{(z)}(j)
\]
\[
\beta_t^{(h)}(i) = \sum_{j} a_{ij} b_j(o_t) \cdot (\beta_{t+1}^{(h)}(j) + \beta_{t+1}(j) \log \frac{a_{ij} b_j(o_t)}{c_{t+1}}),
\]
where the normalization constants \( c_t \) are computed in the forward pass.

A. HMM entropy computation using ESRFB

If the summation of the global kernel \((77)\) is performed over the whole sequence
\[
\bigoplus_s u(s) = \bigoplus_s \left( p(s|o), p(s|o) \log p(s|o) \right),
\]
the \( z \) and \( h \) parts of the sum reduce to
\[
\left( \bigoplus_s u(s) \right)^{(z)} = \sum_s p(s|o) = 1,
\]
\[
\left( \bigoplus_s u(s) \right)^{(h)} = \sum_s p(s|o) \log p(s|o) = -H(S|o)
\]
The \( h \) part of the sum corresponds to the HMM entropy and it can be found as a solution of the normalization problem
\[
\left( \bigoplus_s u(s) \right)^{(h)} = \sum_s \alpha_T^{(h)}(s_T)
\]
using only the forward pass, according to equations \((85)-(86), (90)-(92)\), as follows.
1) Initialization: For \( j = 1, \ldots, N \) set:
\[
c_0 = \sum_{j=1}^{N} \pi_j b_j(o_0), \quad \alpha_0^{(z)}(j) = \frac{\pi_j b_j(o_0)}{c_0}, \quad \alpha_0^{(h)}(j) = \frac{\pi_j b_j(o_0)}{c_0} \log \frac{\pi_j b_j(o_0)}{c_0}.
\] (109)

2) Induction: For \( 1 \leq t \leq T, 1 \leq j \leq N \) compute
\[
c_t = \sum_{i=1}^{N} \alpha_{t-1}^{(z)}(i) a_{jj} b_j(o_t),
\] (111)
\[
\alpha_t^{(z)}(j) = \sum_{i=1}^{N} a_{ij} b_j(o_t) \cdot \alpha_{t-1}^{(z)}(i),
\] (112)
\[
\alpha_t^{(h)}(j) = \sum_{i=1}^{N} a_{ij} b_j(o_t) \cdot \alpha_{t-1}^{(h)}(i) + \alpha_{t-1}^{(z)}(i) \log \frac{a_{ij} b_j(o_t)}{c_t},
\] (113)

3) Termination: Terminate algorithm with summations:
\[
H(S|o) = - \sum_{j=1}^{N} \alpha_T^{(h)}(j),
\] (114)

The ESRFB algorithm runs in \( O(N^2 T) \) time using \( O(N) \) space as in Hernando et al.’s algorithm. Moreover, both algorithms recursively compute the forward probability
\[
\alpha_t^{(z)}(s_t) = p(s_t|o_0:t),
\] (115)
The difference in two algorithms is in the second quantity which is computed - in Hernando et al.’s algorithm it is the intermediate entropy
\[
H_t(s_t) = H(S_{0:t-1}|s_t, o_{0:t}) = - \sum_{s_{t-1} \in \mathcal{T}} p(s_{0:t-1}|s_t, o_{0:t}) \log p(s_{0:t-1}|s_t, o_{0:t}),
\] (116)
while in the ESRFB it is the \( h \)-part of the forward vector:
\[
\alpha_t^{(h)}(s_t) = \sum_{s_{t-1} \in \mathcal{T}} p(s_{0:t}|o_{0:t}) \log p(s_{0:t}|o_{0:t}),
\] (117)
The relation between the quantities
\[
\alpha_t^{(h)} = \alpha_t^{(z)} H_t(s_t) + \alpha_t^{(z)} \log \alpha_t^{(z)}
\] (118)
can easily be derived by use of the elementary probability transformations.

Furthermore, from HMM joint probability factorization we can derive the Markov properties
\[
p(o_t|s_t, s_{t-1}, o_{0:t-1}) = p(o_t|s_t),
\] (119)
\[
p(s_{t-1}|s_t, o_{0:t-1}) = p(s_{t-1}|s_t),
\] (120)
which imply the following equalities:
\[
\frac{a_{s_{t-1}s_t} b_{s_t}(o_t)}{c_t} = \frac{p(s_t|s_{t-1}) p(o_t|s_t)}{p(o_t|o_{0:t-1})} = \frac{p(s_{t-1}|s_t) \cdot \alpha_t^{(z)}(s_t)}{\alpha_{t-1}^{(z)}(s_{t-1})},
\] (121)
where \( \hat{p}_{t-1}(s_t|s_{t-1}) = p(s_{t-1}|s_t, o_{0:t}) \) as defined in the Hernando et al.’s algorithm. Then, the recursive equations for \( H_t(s_t) \) derived by Hernando et al. can also be obtained from the ESRFB algorithm by substituting (121) and (118) in recursive equations for \( \alpha_t^{(h)} \) in ESRFB algorithms, which give us the close relation between two algorithms.

B. HMM subsequence constrained entropy computation using ESRFB

If the summation of the global kernel is performed over a subsequence \( s_{-i:r} \)
\[
\bigoplus_{s_{-i:r}} u(s) = \bigoplus_{s_{-i:r}} \left( p(s|o) \cdot p(o|s) \log p(s|o) \right).
\] (122)
the \( z \) and \( h \) parts of the sum are
\[
\left( \bigoplus_{s_{-i:r}} u(s)^{(z)} \right) = p(s_{i:r})
\] (123)
\[
\left( \bigoplus_{s_{-i:r}} u(s)^{(h)} \right) = - H(S_{i:r}, s_{i:r}|o).
\] (124)
The \( h \) part of the sum corresponds to the HMM subsequence constrained entropy and it can be found as a solution of the marginalization problem
\[
\psi_{i:r}(s_{i:r}) = \alpha(s) \otimes \bigotimes_{i=1}^{r} u_i(s_{i-1}, s_i) \otimes \beta_r(s_r).
\] (125)
The \( z \) and \( h \) parts can be found using the definition for the entropy semiring operations:
\[
\left( \bigoplus_{s_{-i:r}} u(s)^{(z)} \right) = \alpha_i^{(z)}(s_i) \beta_i^{(z)}(s_i) \bigotimes_{i=1}^{r} z_i(s_{i-1}, s_i),
\] (126)
\[
\left( \bigoplus_{s_{-i:r}} u(s)^{(h)} \right) = \bigotimes_{i=1}^{r} z_i(s_{i-1}, s_i) \cdot \left( \alpha_i^{(z)}(s_i) \beta_i^{(z)}(s_i) + \alpha_i^{(h)}(s_i) \beta_i^{(z)}(s_i) + \alpha_i^{(z)}(s_i) \beta_i^{(z)}(s_i) \bigotimes_{j=1}^{r} h_j(s_j-1, s_j) \right)
\] (127)
To compute the \( h \)-part of the marginal, we need \( l \)-th forward and \( r \)-th backward vectors. The \( l \)-th forward vector can be computed by ESR forward algorithm using recursive equations (109)-(113). However, the recursive steps (133)-(113) for the normalization constants \( c_t \) and the \( z \) part of forward vectors should be performed for all \( t \), because the normalization constants \( c_t, r < t < T \) should be available in the backward pass. Once the normalization constants are computed, the backward pass can be performed according to the equations (99), (103)-(104), and, after that, we can compute the subsequence constrained entropy using the equalities (126)-(139) and (18). The algorithm follows.

1) Forward initialization: For \( j = 1, \ldots, N \) set:
\[
c_0 = \sum_{j=1}^{N} \pi_j b_j(o_0), \quad \alpha_0^{(z)}(j) = \frac{\pi_j b_j(o_0)}{c_0},
\] (128)
\[
\alpha_0^{(h)}(j) = \frac{\pi_j b_j(o_0)}{c_0} \log \frac{\pi_j b_j(o_0)}{c_0}.
\] (129)
2) Full forward recursion: For \( 1 \leq t \leq l, 1 \leq j \leq N \) compute

\[
c_t = \sum_{i=1}^{N} \sum_{i=1}^{N} \alpha_{t-1}^{(z)}(i) a_{ij} b_j(o_t)
\]  
  \quad (130)

\[
\alpha_t^{(z)}(j) = \sum_{i=1}^{N} a_{ij} b_j(o_t) c_t \cdot \alpha_{t-1}^{(z)}(i)
\]  
  \quad (131)

\[
\alpha_t^{(h)}(j) = \sum_{i=1}^{N} a_{ij} b_j(o_t) c_t . \alpha_{t-1}^{(z)}(i)
\]  
  \quad (132)

3) Forward z-part recursion: For \( l + 1 \leq t \leq T, 1 \leq j \leq N \) compute

\[
c_t = \sum_{i=1}^{N} \sum_{i=1}^{N} \alpha_{t-1}^{(z)}(i) a_{ij} b_j(o_t)
\]  
  \quad (133)

\[
\alpha_t^{(z)}(j) = \sum_{i=1}^{N} a_{ij} b_j(o_t) c_t \cdot \alpha_{t-1}^{(z)}(i)
\]  
  \quad (134)

4) Backward initialization: For \( j = 1, \ldots, N \) set:

\[
\beta_T^{(z)}(j) = 1, \quad \beta_T^{(h)}(j) = 0.
\]  
  \quad (135)

5) Backward recursion: For \( T - 1 \geq t \geq r, 1 \leq j \leq N \) compute

\[
\beta_t^{(z)}(i) = \sum_j a_{ij} b_j(o_t) c_{t+1} \beta_{t+1}^{(z)}(j)
\]  
  \quad (136)

\[
\beta_t^{(h)}(i) = \sum_j a_{ij} b_j(o_t) c_{t+1} \beta_{t+1}^{(z)}(j)
\]  
  \quad (137)

6) Termination: For \( l \leq t \leq r, 1 \leq s_t \leq N \), compute the subsequence constrained entropy:

\[
p(s_{t:r}) = \alpha_t^{(z)}(s_t) \beta_t^{(z)}(s_t) \prod_{i=t+1}^{r} a_{s_{i-1} s_t} b_{s_i}(o_t) c_t
\]  
  \quad (138)

\[
-H(S_{t:r}, s_{t:r} | o) = \sum_{i=t+1}^{r} a_{s_{i-1} s_t} b_{s_i}(o_t)
\]  
  \quad (139)

\[
\left( \alpha_t^{(z)}(s_t) \beta_t^{(h)}(s_t) + \alpha_t^{(h)}(s_t) \beta_t^{(z)}(s_t) + \alpha_t^{(z)}(s_t) \beta_t^{(z)}(s_t) \sum_{j=t+1}^{r} h_j(s_{j-1}, s_j) \right)
\]  

\[
H(S_{t:r} | s_{t:r}, o) = \frac{H(S_{t:r}, s_{t:r} | o) + \log p(s_{t:r} | o)}{p(s_{t:r} | o)}
\]  
  \quad (140)

The time complexity of the algorithm is \( O(N^2 T + N^{r-1}) \), where \( O(N^2 T) \) is for the forward-backward recursion, and \( O(N^{r-1}) \) for the termination phase, which is the same time complexity as in Mann-MacCallum's algorithm.

On the other hand, full forward recursion phase can be realized in \( O(N^2 t) \) time and in fixed size memory \( O(N) \), since \( \alpha_{t-1}^{(z)}, \alpha_{t-1}^{(h)} \) and \( c_{t-1} \) can be deleted after having been used for the computation of \( \alpha_t^{(z)}, \alpha_t^{(h)} \) and \( c_t \). Similarly, the forward z-part recursion and backward pass requires \( O(N) \) space. Only additional space depending on the sequence length \( O(T - t) \) should be available for normalization constants in the forward z-part recursion phase, since they should be available in the backward and termination phases. Finally, regarding \( O(N^{r-1}) \) space required for storing the results in the termination phase, the total memory complexity is \( O(T - l + N^{r-1}) \), which slightly increases with \( T \) then \( O(N T + N^{r-1}) \), as required by Mann-MacCallum's algorithm.

VI. CONCLUSION

This paper proposes a new algorithm for memory efficient computation of the HMM entropy and subsequence constrained entropy when the observation sequence is given. The algorithm is called Entropy Semiring Forward-backward (ERSFB) since it is based on forward-backward recursion over the entropy semiring in the same manner as in our previous paper [7].

ERSFB has the same time complexity as a previously developed algorithm for subsequence constrained HMM entropy computation developed by Mann and MacCallum [9], but with lower memory requirements. It is also applicable to state sequence entropy computation running with the same time and memory complexity as the recursive algorithm proposed by Hernando et al. [6]. In addition, we have shown how the recursive equations in Hernando et al.’s algorithm can be derived from the ESRFB recursive equations.

REFERENCES

[1] S.M. Aji and R.J. McEliece. The generalized distributive law. Information Theory, IEEE Transactions on, 46(2):325 –343, mar. 2000.

[2] Christopher M. Bishop. Pattern Recognition and Machine Learning (Information Science and Statistics). Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[3] Corinna Cortes, Mehryar Mohri, Ashish Rastogi, and Michael Riley. On the computation of the relative entropy of probabilistic automata. Int. J. Found. Comput. Sci., 19(1):219–242, 2008.

[4] Thomas M. Cover and Joy A. Thomas. Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing). Wiley-Interscience, 2006.

[5] Jason Eisner. Parameter estimation for probabilistic finite-state transducers. In Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, ACL ’02, pages 1–8, Morristown, NJ, USA, 2002. Association for Computational Linguistics.

[6] D. Hernando, V. Crespi, and G. Cybenko. Efficient computation of the hidden markov model entropy for a given observation sequence. Information Theory, IEEE Transactions on, 51(7):2681 – 2685, 2005.

[7] Velimir M. Ilic, Miomir S. Stankovic, and Branimir T. Todorovic. Entropy message passing. IEEE Transactions on Information Theory, 57(1):219–242, 2011.

[8] Wael Khreich, Eric Granger, Ali Miri, and Robert Sabourin. On the memory complexity of the forward-backward algorithm. Pattern Recognition Letters, 31(2):91–99, 2010.

[9] Gideon S. Mann and Andrew McCallum. Efficient computation of entropy gradient for semi-supervised conditional random fields. In Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers on XX, NAACL ’07, pages 109–112, Morristown, NJ, USA, 2007. Association for Computational Linguistics.

[10] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In Proceedings of the IEEE, pages 257–286, 1989.