Recurrent Sum-Product-Max Networks for Decision Making in Perfectly-Observed Environments

Hari Teja Tatavarti CONTACTME.HARITEJA@UGA.EDU

Prashant Doshi PDOSHI@UGA.EDU

Layton Hayes LAYTON.HAYES25@UGA.EDU

Institute for AI, University of Georgia, Athens, GA 30602

Abstract

Recent investigations into sum-product-max networks (SPMN) that generalize sum-product networks (SPN) offer a data-driven alternative for decision making, which has predominantly relied on handcrafted models. SPMNs computationally represent a probabilistic decision-making problem whose solution scales linearly in the size of the network. However, SPMNs are not well suited for *sequential* decision making over multiple time steps. In this paper, we present recurrent SPMNs (RSPMN) that learn from and model decision-making data over time. RSPMNs utilize a template network that is unfolded as needed depending on the length of the data sequence. This is significant as RSPMNs not only inherit the benefits of SPMNs in being data driven and mostly tractable, they are also well suited for sequential problems. We establish conditions on the template network, which guarantee that the resulting SPMN is valid, and present a structure learning algorithm to learn a sound template network. We demonstrate that the RSPMNs learned on a testbed of sequential decision-making data sets generate MEUs and policies that are close to the optimal on perfectly-observed domains. They easily improve on a recent batch-constrained reinforcement learning method, which is important because RSPMNs offer a new model-based approach to offline reinforcement learning.

Keywords: machine learning; sequential decision making; tractable probabilistic models; batch RL.

Appendix A. Proofs

Definition 1 (Template network) *A template network is a directed acyclic graph with r root nodes and at least $n+1$ leaf nodes where n is the number of state variables and there is one utility function. The root nodes form a set of interface nodes Ir. The leaf nodes in the network hold the distributions over the random state variables $X_1, X_2, \ldots, X_n$, hold constant values as utility nodes, or are latent interface nodes. The root interface nodes and interior nodes can either be sum, product, or max nodes. Let L denote the set of leaf latent interface nodes. Each latent node in L is related to a root interface node in Ir of the template network through a bijective mapping f such that $f : L \rightarrow Ir$.*

The bijective mappings can be seen as time delay edges that link latent interface nodes at time step t with root interface nodes at $t+1$, thereby enabling recurrence of the template. The scope of any leaf latent node is itself. But, the scope changes when the template network is unfolded. The scope of a latent node of a template network in time step t is related to the scope of a root interface node of the template network at time step $t+1$. More formally, for any pair of latent nodes $l_i^t, l_j^t \in L$, let $f(l_i^t) = ir_i^{t+1}, f(l_j^t) = ir_j^{t+1}$, where $ir_i^{t+1}, ir_j^{t+1} \in Ir$, then $\Big(scope(ir_i^{t+1}) = scope(ir_j^{t+1})\Big) \Rightarrow \Big(scope(l_i^t) = scope(l_j^t)\Big)$.

Definition 2 (Top network) *A top network is a rooted directed acyclic graph consisting of sum and product nodes, and whose leaves are the latent interface nodes. Edges from a sum node are weighted as in a SPN*

Definition 3 (Soundness of the template) *A template network is sound iff all sum nodes in the template are sum-complete as defined in SPN; all product nodes in the template are decomposable as defined in SPN; all max nodes in the template are max-unique and max-complete as defined in SPMN; the scope of all the root interface nodes in Ir is the same, i.e., $scope(ir_i) = scope(ir_j) \;\; \forall\, ir_i, ir_j \in Ir$; and, the scopes of the leaf latent interface nodes in L are related to that of the mapped root interface nodes in Ir.*

Theorem 1 (Validity of RSPMN) *If, (a) in the top network, all sum nodes are complete and product nodes are decomposable, i.e., top network is valid, and (b) the template network is sound as defined in Def. 3, then the SPMN formed by interfacing the top network and the template network unfolded an arbitrary number of times as needed is valid.*

Proof We sketch a proof by induction. By assumption,

- In the top network, all sum nodes are complete and product nodes are decomposable i.e. top network is valid
- The template network is sound, i.e., all sum nodes are complete, product nodes are decomposable, max nodes are complete and unique.

Base case: We prove that the RSPMN formed by interfacing a top network and a single template network is valid. The relation between scopes of leaf latent interface nodes in the top network with the root interface nodes Ir of template network can be inferred from bijective mapping f as,

$$\begin{aligned} &scope(ir_i) = scope(ir_j) \Rightarrow scope(l_i^{top}) = scope(l_j^{top}), \\ &(l_i^{top}, l_j^{top}) \in L, (ir_i, ir_j) \in Ir, f(l_i) \rightarrow ir_i, f(l_j) \rightarrow ir_j \end{aligned} \tag{1}$$

Since template network is sound,

$$scope(ir_i) = scope(ir_j), \forall (ir_i, ir_j) \in Ir \tag{2}$$

From 1 and 2, the scopes of leaf latent interface nodes of top network become,

$$scope(l_i) = scope(l_j), \forall (l_i, l_j) \in L \tag{3}$$

This means that all the leaf latent interface nodes in top network have same scope. Under this condition and assumption, all the sum nodes of the top network are complete and product nodes are decomposable.

Next, the template network is sound. This means the scopes of all leaf latent interface nodes of template network are same because,

$$scope(ir_i) = scope(ir_j), \forall (ir_i, ir_j) \in Ir \tag{4}$$

From bijective mapping $f(L) \rightarrow Ir$, we can infer

$$\begin{aligned} &scope(ir_i) = scope(ir_j) \Rightarrow scope(l_i) = scope(l_j) \\ &(ir_i, ir_j) \in Ir, (l_i, l_j) \in L \end{aligned} \tag{5}$$

From 4 and 5, the scopes of leaf latent interface nodes of template network become,

$$scope(l_i) = scope(l_j), \forall (l_i, l_j) \in L \tag{6}$$

Under this condition and soundness of template, all sum nodes of template are complete, product nodes are decomposable and max nodes are complete and unique.

Now, when the top network is interfaced with a single template network, the scopes of leaf latent interface nodes of top network change based on relation with root interface nodes Ir at $t = 0$ as below,

$$scope(ir_i^0) = scope(ir_j^0) \Rightarrow scope(l_i^{top}) = scope(l_j^{top}), \tag{7}$$
$$(l_i^{top}, l_j^{top}) \in L, (ir_i^{t+1}, ir_j^{t+1}) \in Ir, f() \rightarrow ir_i, f(L_j) \rightarrow ir_j$$

From 4 we have,

$$scope(ir_i^0) = scope(ir_j^0), \forall (ir_i^0, ir_j^0) \in Ir \tag{8}$$

From 7 and 8,

$$scope(l_i^{top}) = scope(l_j^{top}), \forall (l_i^{top}, l_j^{top}) \in L \tag{9}$$

The condition from 9 is equivalent to the condition from 3. This means the scopes of leaf latent interface nodes of top network have not changed after interfacing with the template network. So, all the sum nodes of top network are complete and product nodes are decomposable even after interfacing with template network. Since no scope is changed in template network after interfacing, all sum nodes are complete, product nodes are decomposable and max nodes are complete and unique in the template network. Therefore the SPMN formed after interfacing top network with single template network is valid.

Induction hypothesis: Let us assume that the SPMN formed after interfacing a top network and the template repeated t times is valid, i.e., all sum nodes are complete, product nodes are decomposable and max nodes are complete and unique. Let this SPMN be R

Inductive step: We now prove that an SPMN formed by interfacing one more template network (template network repeated $(t+1)$ times in total) with R is a valid SPMN.

Since the template network is sound, as we have shown in 4, 5 and 6, we can show that for template at $t+1$,

$$scope(ir_i^{t+1}) = scope(ir_j^{t+1}), \forall (ir_i^{t+1}, ir_j^{t+1}) \in Ir \tag{10}$$
$$scope(l_i^{t+1}) = scope(l_j^{t+1}), \forall (l_i^{t+1}, l_j^{t+1}) \in L \tag{11}$$

and for template at t,

$$scope(ir_i^{t}) = scope(ir_j^{t}), \forall (ir_i^{t}, ir_j^{t}) \in Ir \tag{12}$$
$$scope(l_i^{t}) = scope(l_j^{t}), \forall (l_i^{t}, l_j^{t}) \in L \tag{13}$$

When the template at t is interfaced with the template at $t+1$, the scopes of leaf latent interface nodes of template at t relate to root interface nodes of template at $t+1$ as follows,

$$scope(ir_i^{t+1}) = scope(ir_j^{t+1}) \Rightarrow scope(l_i^{t}) = scope(l_j^{t}), \tag{14}$$
$$(l_i^{t}, l_j^{t}) \in L, (ir_i^{t+1}, ir_j^{t+1}) \in Ir, f() \rightarrow ir_i, f(L_j) \rightarrow ir_j$$

From 10 and 14 we have,

$$scope(l_i^{t}) = scope(l_j^{t}), \forall (l_i^{t}, l_j^{t}) \in L \tag{15}$$

The condition from 15 is equivalent to the condition from 13. This means the scopes of leaf latent interface nodes of template network at t have not changed after interfacing with the template network at $t+1$. From inductive hypothesis, SPMN R is valid. Since there is no change in scopes of any of the nodes in R after interfacing with template at $t+1$, all sum nodes are complete, product nodes are decomposable and max nodes are complete and unique in R. Since no scope is changed in template network at $t+1$ after

interfacing, all sum nodes are complete, product nodes are decomposable and max nodes are complete and unique in the template network at $t+1$. So, all sum nodes are complete, product nodes are decomposable and max nodes are complete and unique in the SPMN formed after interfacing R with template network at $t+1$. Therefore, the SPMN formed after interfacing R with one more template network (template repeated $t+1$ times in total) is valid.

■

Appendix B. Algorithms

Algorithm 1 presents the four main steps involved in learning the RSPMN from data which we refer to as LEARNRSPMN

Algorithm 1: LEARNRSPMN

Input : Dataset $\langle \tau^0, \tau^1, \ldots, \tau^T \rangle_e$ where $e = 1, 2, \ldots, E$; Partial Order $P^{\prec}$
Output : top network, template network

$\mathcal{S}^{t=2} \leftarrow$ Run LEARNSPMN over wrapped 2-time step data $\langle \tau^0, \tau^1 \rangle_{e'} \quad e' = 1, 2, \ldots, T \times E$
Create top network and set of root interface nodes Ir from $\mathcal{S}^{t=2}$
Create initial template network from Ir and $\mathcal{S}^{t=2}$
Revise initial template using sequence data to obtain the final template

[Step 1] Learn an SPMN from 2-time step data The first step of LEARNRSPMN is to use LearnSPMN algorithm to learn a valid SPMN, $\mathcal{S}^{t=2}$, from 2-time step data by wrapping the data set with $P^{\prec}$ as the partial order. This process is shown in algorithm

Algorithm 2: SPMN from 2-time step data

input : Dataset: $\langle \tau^0, \tau^1, \ldots, \tau^T \rangle_e$ where $e = 1, 2, \ldots, E$, Partial Order: $P^{\prec}$
output : SPMN from 2-time step data: $\mathcal{S}^{t=2}$

$w^0 \leftarrow$ Empty, $w^1 \leftarrow$ Empty
for e *in* $1, 2, \ldots, E$ **do**
    for t *in* $0, 1, \ldots T-1$ **do**
        $w^0 \leftarrow w^0.add(\tau^t)$
        $w^1 \leftarrow w^1.add(\tau^{t+1})$
$W \leftarrow \langle w^0, w^1 \rangle$
$\mathcal{S}^{t=1} \leftarrow$ LEARNSPMN(W, $P^{\prec}$)

[Step 2] Obtain top network and Ir nodes First we extract a 1-time step network $\mathcal{S}^{t=1}$ from $\mathcal{S}^{t=2}$ and we obtain the nodes in Ir using $\mathcal{S}^{t=1}$. This is shown in Algorithms 3 and 4

[Step 3] Building an initial template Ir from the previous step containing the root interface nodes forms a basis for creating the initial template. The sum node S_L whose children are the set of leaf latent nodes L is added to Ir to obtain initial template structure as shown in Algorithm 5

[Step 4] Learning the final template network By updating the edge weights of the outgoing edges from the sum node S_L whose children are set of leaf latent nodes L, we can learn the final template structure as shown in Algorithm 6.

Algorithm 3: Create top network and set of root interface nodes

input : Two step SPMN: $\mathcal{S}^{t=2}$
output : Top Network $\mathcal{S}_O$, Set of root interface nodes Ir

$Queue \leftarrow \mathcal{S}^{t=2}.root$
while $Queue$ *is not* $\emptyset$ **do**
  $node \leftarrow Queue.dequeue$
  if $node$ *is product* **then**
    for *each child* c *in the set of node's children* C_n **do**
      if c *does not have any variable of* $(I_0, d_1, I_1, d_2, \ldots, I_{m-1}, d_m, I_m, u)^1$ *in its scope* **then**
        Remove c from C_n
One step network $\mathcal{S}^{t=1} \leftarrow$ remaining $\mathcal{S}^{t=2}$
set of nodes seen $E \leftarrow \emptyset$; root interface nodes $Ir \leftarrow \emptyset$
$Queue \leftarrow \mathcal{S}^{t=1}.root$
while $Queue$ *is not* $\emptyset$ **do**
  $node \leftarrow Queue.dequeue$
  if *(*$node$ *is product and all* $(I_0, d_1, I_1, d_2, \ldots, I_{m-1}, d_m, I_m, u)^0$ *are in* $node.scope$*)* **then**
    topProdChildren $C \leftarrow \emptyset$
    `/* c.scope must be a proper subset */`
    if *(each* $c.scope \subset (I_0, d_1, I_1, d_2, \ldots, I_{m-1}, d_m, I_m, u)^1, \forall c \in node.children$*)* **then**
      $C \leftarrow node.children$
      $node.children \leftarrow \emptyset$
    if C *is not* $\emptyset$ **then**
      Create Product node P ; $P.children \leftarrow C$
      $R \leftarrow IrChildren\,(P)$
      latent interface nodes $L \leftarrow \emptyset$
      for *each* $irchildren$ *set* ir_C *in* R **do**
        Create interface root product node ir
        $ir.children \leftarrow ir_C$
        Create latent interface node l with a bijective mapping $f(l) \rightarrow ir$
        $L \leftarrow L \cup l$; $Ir \leftarrow Ir \cup ir$
      Create Sum node S_L with $S_L.children \leftarrow L$
      $node.children \leftarrow node.children \cup S_L$
    else
      for *each child* c *in the set of node's children* C_n **do**
        if c *is not in* E **then**
          Add c to E
          Enqueue c into $Queue$
Top Network $\mathcal{S}_O \leftarrow$ remaining $\mathcal{S}^{t=1}$
Set of root interface nodes Ir

Algorithm 4: make sets of Ir children

input : Product node P

output : set of interface root node children sets

```
1  Function IrChildren(P):
2      if (any c is Sum, ∀c ∈ P.children) then
3          for each c in P.children do
4              stateVars X ← ∅
5              if c is Sum then
                   /* a set of sets                                    */
6                  set of IrChildren sets R ← ∅
7                  for each c_p in c.children do
8                      irChildren set ir_C ← IrChildren(c_p)
9                      R ← R ∪ ir_C
10             else
11                 X ← X ∪ c
12         for each ir_C in R do
13             ir_C ← ir_C ∪ X
14         return R
15     else
16         return {{P}}
```

Algorithm 5: Create initial template network

input : Set of root interface nodes Ir
output : Initial template root interface nodes Ir

$L \leftarrow \emptyset$
for *each* ir *in* Ir **do**
    Create a Latent interface node l and $f(l) \rightarrow ir$
    $L \leftarrow L \cup l$
for *each* ir *in* I **do**
    set of nodes seen $E \leftarrow \emptyset$
    $Queue \leftarrow ir.root$
    while $Queue$ *is not* $\emptyset$ **do**
        $node \leftarrow Queue.dequeue$
        if *(each c is Leaf node,* $\forall c \in node.children$*)* **then**
            Create Sum node S_L `/* bottom sum interface node */`
            $S_L.children \leftarrow L$
            $S_L.weights \leftarrow 1/numOf(S_L.children)$
            if $node$ *is product* **then**
                Add S_L as child of product node
            else
                for *each* l *that is Leaf node in set of* $node.children$ C_n **do**
                    Create Product node P
                    $P.children \leftarrow S_L \cup l$
                    $C_n \leftarrow C_n \setminus l$
                    $C_n \leftarrow C_n \cup P$
        else
            for *each* $c \in node.children$ **do**
                if c *is not in* E **then**
                    Add c to E
                    Enqueue c into $Queue$
Initial template root interface nodes $Ir \leftarrow$ set of root interface nodes Ir

Algorithm 6: Final template network

input : Initial template root interface nodes Ir; Top Network $\mathcal{S}_O$
output : Final template root interface nodes

$T \leftarrow$ length of sequence
$ll^{-1}, ll^0, \ldots, ll^{T-1} \leftarrow$ EmptyList
for *each time step* t *from* $T-1 \ldots 0$ **do**
    if $t = T$ **then**
        Remove leaf latent nodes from Ir
        Assign leaf values $(I_0, d_1, I_1, d_2, \ldots, I_{m-1}, d_m, I_m, u)^t$ to leaves in Ir
        $ll^t \leftarrow$ bottomUpPass(Ir)
    else
        Assign leaf values $(I_0, d_1, I_1, d_2, \ldots, I_{m-1}, d_m, I_m, u)^t$ to leaves in Ir
        Assign Ir values from ll^{t+1} to latent leaves in Ir
        $ll^t \leftarrow$ bottomUpPass(Ir)
Assign Ir values from ll^0 to latent leaves of $\mathcal{S}_O$
$ll^{-1} \leftarrow$ bottomUpPass($\mathcal{S}_O$)
for *each time Step* t *from* $-1 \ldots m-1$ **do**
    if $t = -1$ **then**
        TopDownPass($\mathcal{S}_O$)
        $l \leftarrow$ leaf latent interface node reached
    else
        $I_l \leftarrow$ root interface node corresponding to l
        TopDownPass(I_l)
        Increment counts on outgoing edges visited on sum nodes
        $l \leftarrow$ leaf latent interface node reached
    Update weights on sum node S_L whose children are L in Ir using counts
    Drop branches with zero counts on S_L
    Final template root interface nodes $Ir \leftarrow Ir$