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 Def.* **??***; all product nodes in the template are decomposable as defined in Def.* **??***; all max nodes in the template are max-unique and max-complete as defined in Defs.* **??** *and* **??***; the scope of all the root interface nodes in Ir is the same, i.e., $scope(ir_i) = scope(ir_j) \quad \forall\, ir_i, ir_j \in Ir$; 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 invariant, 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 scopes of latent interface leaf nodes in the top network are related to interface root nodes I of template network by definition of top network 2,

$$\begin{aligned} &scope(I_i) = scope(I_j) \Rightarrow scope(L_i^{top}) = scope(L_j^{top}), \\ &(L_i^{top}, L_j^{top}) \in L, (I_i, I_j) \in I, f(L_i) \rightarrow I_i, f(L_j) \rightarrow I_j \end{aligned} \tag{1}$$

Since template network is invariant,

$$scope(I_i) = scope(I_j), \forall (I_i, I_j) \in I \tag{2}$$

From 1 and 2, the scopes of latent interface leaf 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 latent interface leaf nodes in top network have same scope. Under this condition, all the sum nodes of the top network are complete and product nodes are decomposable by assumption.

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

$$scope(I_i) = scope(I_j), \forall (I_i, I_j) \in I \tag{4}$$

$$\begin{aligned} &scope(I_i) = scope(I_j) \Rightarrow scope(L_i) = scope(L_j) \\ &(I_i, I_j) \in I, (L_i, L_j) \in L, f(L) \rightarrow I \end{aligned} \tag{5}$$

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

$$scope(L_i) = scope(L_j), \forall (L_i, L_j) \in L \tag{6}$$

Under this condition and invariance 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 latent interface leaf nodes behave according to definition **??** as,

$$\begin{aligned} &scope(I_i^0) = scope(I_j^0) \Rightarrow scope(L_i^{top}) = scope(L_j^{top}), \\ &(L_i^{top}, L_j^{top}) \in L, (I_i^{t+1}, I_j^{t+1}) \in I, f(L_i) \rightarrow I_i, f(L_j) \rightarrow I_j \end{aligned} \tag{7}$$

From 4 we have,

$$scope(I_i^0) = scope(I_j^0), \forall (I_i^0, I_j^0) \in I \tag{8}$$

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 latent interface leaf 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 RSPMN formed after interfacing top network with single template network is valid.

Induction hypothesis: Let us assume that the RSPMN 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 RSPMN be R

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

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

$$scope(I_i^{t+1}) = scope(I_j^{t+1}), \forall (I_i^{t+1}, I_j^{t+1}) \in I \tag{10}$$

$$scope(L_i^{t+1}) = scope(L_j^{t+1}), \forall (L_i^{t+1}, L_j^{t+1}) \in L \tag{11}$$

and for template at t,

$$scope(I_i^t) = scope(I_j^t), \forall (I_i^t, I_j^t) \in I \tag{12}$$

$$scope(L_i^t) = scope(L_j^t), \forall (L_i^t, L_j^t) \in L \tag{13}$$

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

$$scope(I_i^{t+1}) = scope(I_j^{t+1}) \Rightarrow scope(L_i^t) = scope(L_j^t), \tag{14}$$
$$(L_i^t, L_j^t) \in L, (I_i^{t+1}, I_j^{t+1}) \in I, f(L_i) \rightarrow I_i, f(L_j) \rightarrow I_j$$

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 latent interface leaf nodes of template network at t have not changed after interfacing with the template network at $t+1$. From inductive hypothesis, RSPMN 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 RSPM formed after interfacing R with template network at $t+1$. Therefore, the RSPMN formed after interfacing R with one more template network (template repeated $t+1$ times in total) is valid.

■

Appendix B. Algorithms

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

1 $\mathcal{S}^{t=2} \leftarrow$ Run LEARNSPMN over wrapped 2-time step data $\langle \tau^0, \tau^1 \rangle_e \quad e = 1, 2, \ldots, T \times E$
2 Create top network and set of root interface nodes Ir from $\mathcal{S}^{t=2}$
3 Create initial template network from Ir and $\mathcal{S}^{t=2}$
4 Revise initial template using sequence data to obtain the final template

Algorithm 2: Create top network and set of interface root nodes

ht
input : Two step SPMN: N_W
output : Top Network O, Set of Interface Root Nodes I
$Queue \leftarrow N_W.root$
while $Queue$ *is not* $\emptyset$ **do**
  $node \leftarrow Queue.dequeue$
  if $node$ *is product* **then**
    for *each child* c_j *in the set of node's children* C_n **do**
      if c_j *does not have any variable of* $\langle S_0^{w_0}, \ldots A_k^{w_0}, \ldots U_n^{w_0} \rangle$ *in its scope* **then**
        Remove c_j from C_n
One step network $N_{w_0} \leftarrow$ remaining N_W
set of nodes seen $E \leftarrow \emptyset$; Interface root nodes $I \leftarrow \emptyset$
$Queue \leftarrow N_{w_0}.root$
while $Queue$ *is not* $\emptyset$ **do**
  $node \leftarrow Queue.dequeue$
  if *(*$node$ *is product and all* $\langle S_0^{w_0}, \ldots A_k^{w_0}, \ldots U_n^{w_0} \rangle$ *are in* $node.scope$*)* **then**
    set of interface children $C \leftarrow \emptyset$
    `/* ` $c_j.scope$ `must be a proper subset */`
    if *(each* $c_j.scope \subset \langle S_0^{w_0}, \ldots A_k^{w_0}, \ldots U_n^{w_0} \rangle, \forall c_j \in node.children$*)* **then**
      $C \leftarrow node.children$; $node.children \leftarrow \emptyset$
    if C *is not* $\emptyset$ **then**
      Create Product node P_c ; $P_c.children \leftarrow C$
      $R \leftarrow$`setOfInterfaceRootChildrenSets(`P_c`)`
      latent interface nodes $L \leftarrow \emptyset$
      for *each root children set* C *in* R **do**
        Create interface root product node P_I ; $P.children \leftarrow C$
        Create latent interface node l with a bijective mapping $f(l) \rightarrow P_I$
        $L \leftarrow L \cup l$; $I \leftarrow I \cup P_I$
      Create Sum node S_L with $S_L.children \leftarrow L$ $node.children \leftarrow node.children \cup S_L$
    else
      for *each child* c_j *in the set of node's children* C_n **do**
        if c_j *is not in* E **then**
          Add c_j to E ; Enqueue c_j into $Queue$
Top Network $O \leftarrow$ remaining N_{w_0} ; Set of Interface root nodes I

Algorithm 3: make set of interface root node children sets

input : Product node P_c

output : set of interface root node children sets

```
1  Function setOfInterfaceRootChildrenSets(P_c):
2      if (any c_j ∈ Sum, ∀c_j ∈ P_c.children) then
3          for each c_j in P_c.children do
4              stateVars S ← ∅
5              if c_j is Sum then
                   /* a set of sets                                   */
6                  set of interfaceRootChildren sets R ← ∅
7                  for each c_p in c_j.children do
8                      interfaceChildrenSet F_{c_p} ← setOfInterfaceRootChildrenSets(c_p)
9                      R ← R ∪ F_{c_p}
10             else
11                 S ← S ∪ c_j
12         for each C_f in R do
13             C_f ← C_f ∪ S
14         return R
15     else
16         return {{P_c}}
```

Algorithm 4: Create initial template network

input : Set of Interface Root Nodes I
output : Initial Template Network T

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

Algorithm 5: Final template network

input : Initial Template Network T; Top Network O

output : Final Template Network T_f

$m \leftarrow$ length of sequence - 1

$ll^-1, ll^0, \ldots, ll^{m-1} \leftarrow$ EmptyList

for *each time step* t *from* $m \ldots 0$ **do**

  if $t = m$ **then**

    $T.leafNodes \leftarrow \langle s_0, \ldots a_k, \ldots u_n \rangle^m$

    $T.latentLeafNodes\ (T.L.likelihood)) \leftarrow 1$

    $ll^m \leftarrow likelihood(T)$

  else

    $T.leafNodes \leftarrow \langle s_0, \ldots a_k, \ldots u_n \rangle^t$

    $T.L.likelihood \leftarrow ll^{(t+1)}(f^{-1}(I))$

    $ll^t \leftarrow likelihood(T)$

$O.L.likelihood \leftarrow ll^0(f^{-1}(I))$

$ll^{-1} \leftarrow likelihood(O)$

for *each time Step* t *from* $-1 \ldots m-1$ **do**

  if $t = -1$ **then**

    $l^t \leftarrow$ TopDown$(O, L|ll^t, updateCounts = True)$

  else

    $I_l \leftarrow f(L = l^{t-1})$

    $l^t \leftarrow$ TopDown$(T.I_l, L|ll^t, updateCounts = True)$

  Final template network $T_f \leftarrow$ UpdateWeights(T)