Recurrent Sum-Product-Max Networks for Multi-Agent Decision Making: A Perspective

Hannah Tawashy
THINC Lab, School of Computing
University of Georgia
Athens, USA
hmtx@uga.edu

Prashant Doshi
THINC Lab, School of Computing
University of Georgia
Athens, USA
pdoshi@uga.edu

ABSTRACT

This position paper presents a novel use of recurrent sum-product-max networks (RSPMN) for modeling multi-agent decision making. RSPMNs are an extension of SPNs, a model that can perform tractable exact inference for sequential decision making. The structure of RSPMNs is learned using a data driven approach. We outline three methods utilizing RSPMNs, ranging from centralized to decentralized multi-agent decision making. In a fully centralized approach, the RSPMN models the joint decision making of all agents. With centralized learning and decentralized decision making, the RSPMN is decomposed into a sub-RSPMN for each agent while still allowing for centralized coordination. In a fully decentralized approach, each agent has its own RSPMN and learns representations of other agents' behaviors. The agent would then use this representation as a part of its own model to make decisions. Limitations of implementing RSPMNs for this purpose are also briefly explored.

KEYWORDS

Sum-Product Networks, Multi-Agent Modeling, Decision Making, Tractability, Scalability, Data Driven

1 INTRODUCTION

Decision making is an essential aspect of many multi-agent environments. This process consists of making predictions of multiple agents' behavior on the basis of available evidence, including past decisions and the state of the system. Agents make decisions based on either their own observations of the environment and reasoning of other agents' actions or the collected observations of all agents. A large number of agents and interactions may cause inference to become exponentially complex, leading to intractable reasoning. Therefore, when designing these multi-agent systems, it is crucial to consider scalability. One approach to modeling these systems are graphical models, a valuable tool which also provides a visual representation of the relationships between agent state, inference, and decisions. The structure of the graph then determines the complexity of performing inference and decision making.

Sum-product networks (SPN) [3] are a graphical model that can perform tractable exact inference for most types of probabilistic inferences. **Sum-product-max networks (SPMN)** [5] are built upon the original SPN to translate the model to decision making. We propose deploying **recurrent-sum-product-max networks (RSPMNs)** [8], an extension of SPMNs that adapt SPMNs to temporal or dynamic environments, to model multi-agent decision making. The max-product message passing algorithm used in SPNs has a complexity that is linear with the size of the network [3]. This fundamental feature lends itself to a tractable multi-agent implementation, particularly in complex systems with a large number of agents. An additional benefit inherited from SPNs is that both their structure and parameters are learned directly from the input data. This is especially useful when neither model specifications nor the environment is available to the agents.

Proc. of the Multiagent Sequential Decision Making Workshop (MSDM 2023) , Zeng, Xiao, Pan, and Doshi (eds.), May 30, 2023, Online, https://northumbria23.github.io/MSDM23/. 2023.

2 BACKGROUND

SPNs are directed acyclic graphical models that utilize sum and product nodes to represent probability distributions across random variables represented in the leaf nodes. The sum nodes model the disjunctive data, while the product nodes capture the dependencies between random variables [3]. A valid network can produce any marginal, joint, or conditional inference over its random variables through at most two passes of this structure.

Structure learning and parameter learning are integral processes in building an SPN. Structure learning uncovers the underlying network structure by discovering the relationships between variables. Parameter learning generates the estimates for the SPN parameters, including the weights on the sum node's children and distributions of the variables. LearnSPN, a popular algorithm for structure and parameter learning, is a top-down approach. Fundamental processes include initialization, partitioning, decomposition, local parameter learning, merging, and pruning [3]. The SPN is incrementally built and minimizes the negative log-likelihood of the training data. This data-driven approach is a central proponent of SPNs. The algorithm continues to iterate until a stopping criterion is met, such as convergence of likelihood or reaching an iteration threshold.

Consistency, completeness, and decomposability are the trinity of properties that allow SPNs to accurately and efficiently perform valid inference. Intuitively, consistency ensures the proper normalization of probabilities, meaning that the sum of weights at each sum node is equal to one. Completeness guarantees that the probability distribution of any combination of the random variables can be computed. Decomposability allows for efficient inference and an interpretable structure, as the SPN can be decomposed into a product of simple distributions. Poon and Domingos [3] provides the technical definitions of these properties.

SPMNs extend SPNs to decision making by introducing max and utility nodes [5]. Max nodes represent action choice through modeling utility using the utility nodes. The output generated by max nodes is the maximum of the possible actions' expected utility values, and is fed upward to the rest of the network [5]. The utility

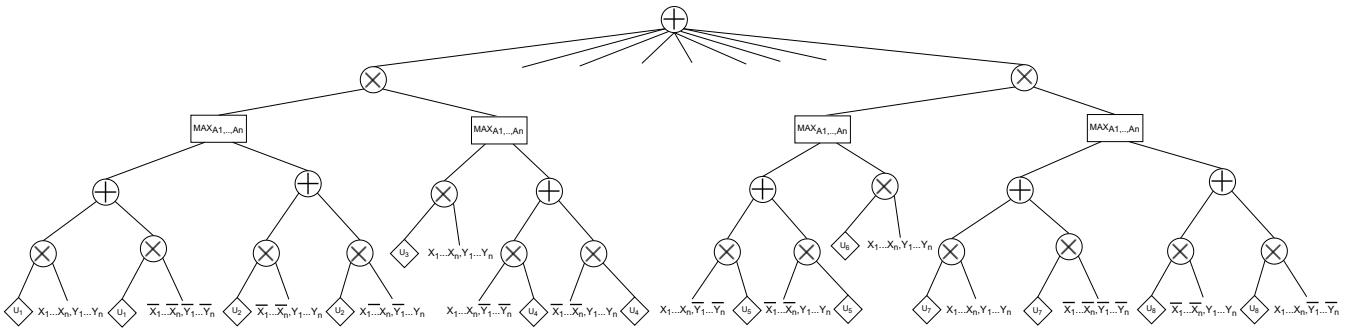

Figure 1: Schematic of an example RSPMN structure with binary random variables that supports centralized learning and joint decision making.

values can be determined by a relevant domain expert, adapting the network to a specific environment context and reward function.

RSPMNs are a further generalization built upon SPMNs, augmenting SPMNs to handle a temporal environment [8]. Inspired by recurrent neural networks, a single time-step template in the RSPMN is unrolled across multiple time steps to obtain the dynamic network. This template is a snapshot of the network structure at an exact time step. The sequence of networks maintains a memory of previous decisions through their recursive connection, as the output of one network becomes the input of the next network. This framework empowers SPNs to model sequential decision making. Decision making is represented as a sequence of max operations which select the highest expected value action(s) considering the random variable values and utilities in each time step.

Previous graphical models that are commonly used for modeling decision making in multi-agent environments are generally intractable [1, 4, 9]. These are classically based on *dynamic influence diagrams (DIDs)* [6], and utilize various approaches to attempt to mitigate their tractability issues. A combination of decentralized designs and approximate inference algorithms trade accuracy for computation time. These explorations are not always successful. RSPMNs, unlike these models, can perform exact inference and decision making that is linear in the size of the network [8]. Additionally, the network structure can be learned directly from given data, rather than relying on a specification of the structure or necessary parameters as in MDPs and DIDs.

3 RSPMNS FOR MULTI-AGENT ENVIRONMENTS

RSPMNs can be used for a spectrum of decision-making approaches in multi-agent environments [7]. In a centralized approach, decision making is represented by a unified agent. This central agent accumulates the information from all agents and utilizes it to generate a joint decision. On the other extreme, a fully decentralized approach (individualistic) distributes decision making amongst the agents [2]. We propose using RSPMNs in three different ways: centralized learning and decision making, centralized learning and decentralized decision making, and fully decentralized learning and decision making.

3.1 Centralized Learning and Decisions

In this fully centralized approach, as seen in Figure 1, the RSPMN is a unified representation. It represents a single joint model of all agents in the environment. The partial ordering, which refers to the organization of variables in the network, is designed such that the centralized nature of the system is captured. The relationships between the variables in the system as a unit are reflected by this order. The leaves of the network receive input data from all agents, including their observations of the current state of the system and preferences. These leaves, representing the random variables modeled by the RSPMN, would be all possible joint observations of the agents for each random variable. The network computes the joint distribution of decisions made by *all agents*, represented by the max nodes. Agent coordination is facilitated by node operations, which pass messages through their edges. This allows the network to learn a joint policy that maps joint observations of all agents to their joint actions.

The previous LearnRSPMN structure and parameter learning algorithm [8] can be utilized almost as is in this approach. The training data contains instances of the agents' observations, actions, and rewards – i.e., values of the corresponding random variables. This includes the joint decisions made by all agents at each time step. This input data is used to train the RSPMN, allowing the network to learn the dependencies between the agent states and joint decisions. The learning process would include optimizing the parameters by minimizing the negative log-likelihood, or other similar loss function, between the actual joint decisions and the prediction made by the network. This approach allows for tractable joint decision making in environments where agents are collaboratively operating on a single task.

3.2 Centralized Learning and Decentralized Decisions

A hybrid approach, as seen in Figure 2, combines centralized learning and decentralized decision making by decomposing the main RSPMN into subnetworks. In this approach, each agent is assigned their own subnetwork with max nodes representing the decision choice of a *single corresponding agent*. The leaves of each subnetwork receive input data of the respective agent's observations of the system state, actions, and reward values. Thus, each subnetwork,

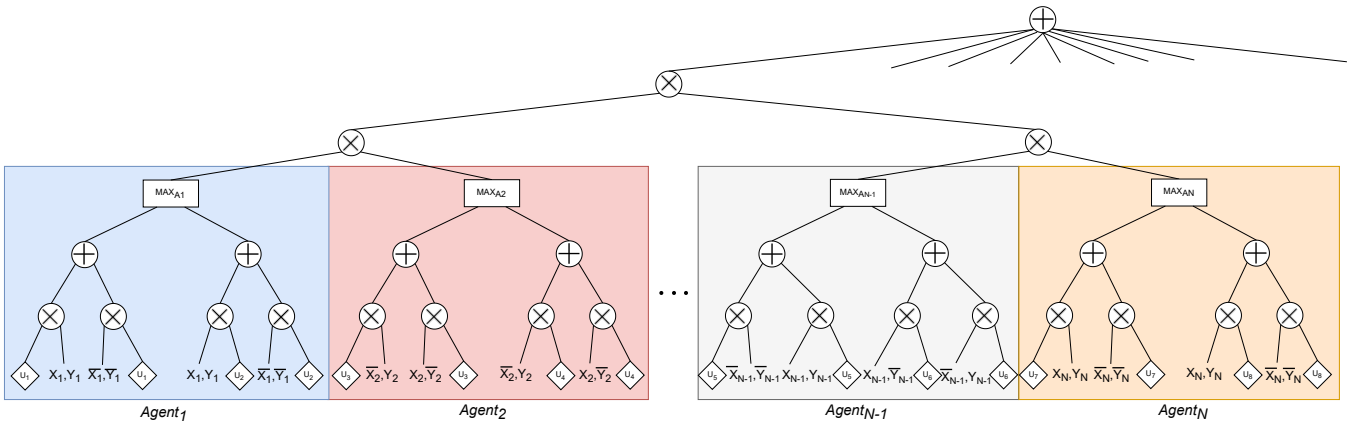

Figure 2: Schematic of an example RSPMN structure that supports centralized learning and decentralized decision making. The other root sum node branch is similar to the product node subnetwork.

and respective agent, models its own set of random variables as leaves to serve as the input point for the agent's training data from its explorations. The partial ordering expresses the dependencies between the variables for each individual agent in a decentralized fashion.

Semantically, the overall RSPMN would still represent all agents in the environment with the difference that each agent's actions are conditioned on its own observations only. The top subnetwork that connects the various agents' models represents a message passing system that could relay the necessary information between agents, such as that required for coordination among them. With this, each agent learns its own policy, which promotes the centralized learning of decentralized decision-making models in conjunction with any needed coordination mechanism.

The decomposition of the RSPMN into subnetworks means that the significant divergence between this approach and the fully centralized approach lies in the semantics of the max nodes. This approach shifts the training input data from joint observations and joint actions across all agents to the observations and actions of each of the agents.

Nonetheless, the entire RSPMN can still be learned as one network using LearnRSPMN, analogously to the centralized method, but we may need to provide a structural template to the learning algorithm in order to clearly separate the agents' subnetworks. We will learn the dependencies between states of each individual agent and their respective decisions, while also still considering potential interactions between the agents. As such, this approach allows the network to recognize individual agents, while still modeling the dependencies between observations and decisions, and between each other, for data-driven and tractable decision making.

3.3 Decentralized Learning and Decisions

Finally, a fully decentralized approach isolates agent learning and decision making entirely. To model this, *each agent* has its own RSPMN (see Figure 3), with the leaves representing the random variables that model the agent's observations, and reward values. Since there are no edge connections between independent RSPMNs, these agents will have to learn a representation of the other agents' decision-making behavior and interpret them to make decisions. Thus, agents require having evidence of how other agents in the environment act. This necessitates adding one or more additional random variable $\boldsymbol{Z}$, as seen in Figure 3, to capture behavior representations of the other agents in the shared system.

The training data consists of the individual agents' observations of its state, actions, reward values as well as some evidence of the other agents' behaviors (for e.g., these could be observations of its actions). This approach may call for modification to structural and parameter learning to integrate a learned representation to incorporate this evidence.

This fully decentralized learning allows agents to learn their own policy on the basis of their own observations. There are no messages passed between agents, so the policy is entirely based on the individual agents' information, which may include a learned representation of other agents's behaviors from its own observations. As such, each agent would have their own representation of the environment. A consideration is that the influence of the other agents' behaviors on the subject agent may be hidden and lose interpretation, running contrary to the benefit of SPNs.

4 CONCLUSION

Across a multitude of domains, systems of multiple agents are prevalent. In realistic environments, the number of agents can cause tractability issues in traditional models. This paper highlights the potential of RSPMNs as an efficient approach to modeling multi-agent decision making. The outlined approaches offer scalable solutions for inference in complex multi-agent environments, with a linear complexity in the size of the network. The network structure is learned directly from the data, setting the model apart from traditional approaches. The three schemes we present allow RSPMNs to model various multi-agent systems, such as partially observable and non-cooperative environments. In future work, we intend to develop the learning algorithms and apply the RSPMNs to multi-agent environments to demonstrate their practicality over existing models in realistic environments.

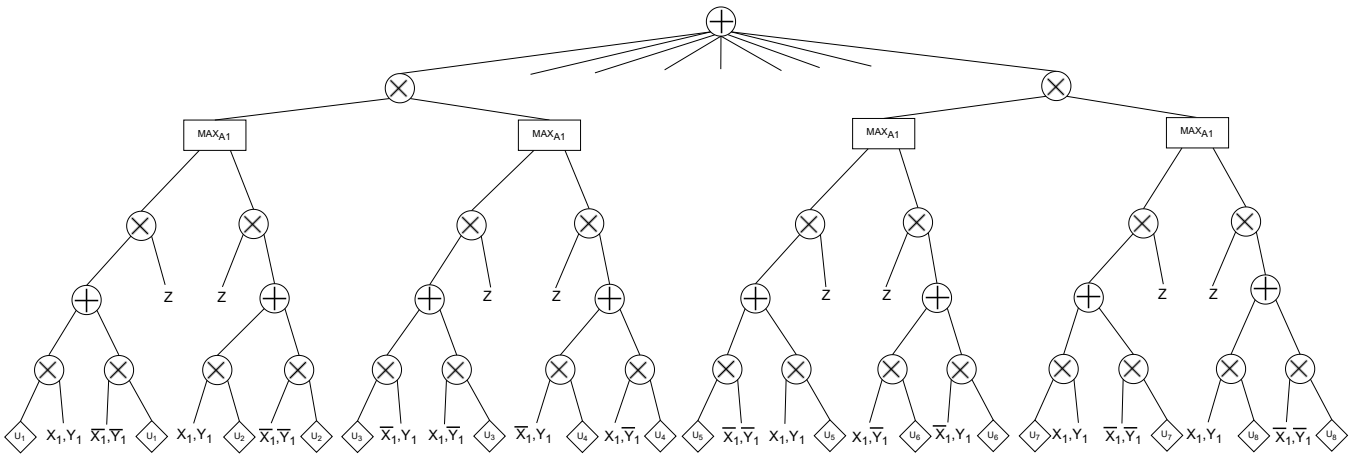

Figure 3: Schematic of an example RSPMN structure that supports decentralized learning and decision making.

Limitations. Although RSPMNs are advantageous due to their tractability and data driven approach, there are a few potential limitations. When considering systems with a large amount of agents, the network can grow unbounded in size. This will not affect the tractability, but will negatively impact run time and memory. This increased computational overhead can make training and maintaining the RSPMN model challenging. Learning the structure of an RSPMN directly from data allows for the network to be finetuned, however finding a suitable training set is not a trivial task. After identifying a data set, it is essential to rigorously analyze the data as the quality of the input is directly related to the quality of the output.

ACKNOWLEDGMENTS

We thank Daniel Redder and Gayathri Anil for reviewing and editing this paper, and Hari Tatavarti for assistance with RSPMNs. This research was supported in part by NSF grant #IIS-1815598.

REFERENCES

[1] Kobi Gal and Avi Pfeffer. 2008. Networks of Influence Diagrams: A Formalism for Representing Agents' Beliefs and Decision-Making Processes. *Journal of Artificial Intelligence Research* 33 (2008), 109–147.

[2] Piotr J. Gmytrasiewicz and Prashant Doshi. 2005. A Framework for Sequential Planning in Multiagent Settings. *Journal of Artificial Intelligence Research* 24 (2005), 49–79.

[3] Pedro Domingos Hoifung Poon. 2011. Sum-product networks: A new deep architecture. In *Proceedings of the Conference on Computer Vision Workshops (ICCV Workshops)*. The Institute of Electrical and Electronics Engineers, Barcelona, Spain, 689–690.

[4] Daphne Koller and Brian Milch. 2001. Multi-agent Influence Diagrams for Representing and Solving Games. In *IJCAI*. 1027–1034.

[5] Prashant Doshi Mazen Melibari, Pascal Poupart. 2016. Sum-Product-Max Networks for Tractable Decision Making. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (ICCV Workshops, Vol. IJCAI-16)*. The Association for the Advancement of Artificial Intelligence, New York City, New York, 1846–1852.

[6] Robert M. Oliver and James Q. Smith (Eds.). 1990. *Influence Diagrams, Belief Nets, and Decision Analysis*. John Wiley & Sons.

[7] Victor Lesser Ping Xuan. 2002. Multi-agent policies: From Centralized Ones to Decentralized Ones. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems*. The Association for Computing Machinery, Bologna, Italy, 1098–1105.

[8] Hari Teja Tatavarti. 2020. *Recurrent Sum-Product-Max Networks for Decision Making in Perfectly-Observed Environments*. Master's thesis. University of Georgia, Athens, Georgia.

[9] Yifeng Zeng and Prashant Doshi. 2012. Exploiting Model Equivalences for Solving Interactive Dynamic Influence Diagrams. *Journal of Artificial Intelligence Research* 43 (2012), 211–255.