Treatment-RSPN: Recurrent Sum-Product Networks for Sequential Treatment Regimes

Adam Dejl
MIT
adamdejl@mit.edu

Harsh Deep
MIT
hdeep@mit.edu

Jonathan Fei
MIT
jyfei@mit.edu

Ardavan Saeedi
Optum Labs*
av.saeedi@gmail.com

Li-wei H. Lehman
MIT, MIT-IBM Watson AI Lab
lilehman@mit.edu

Abstract

Sum-product networks (SPNs) have recently emerged as a novel deep learning architecture enabling highly efficient probabilistic inference. Since their introduction, SPNs have been applied to a wide range of data modalities and extended to time-sequence data. In this paper, we propose a general framework for modelling sequential treatment decision-making behaviour and treatment response using recurrent sum-product networks (RSPNs). Models developed using our framework benefit from the full range of RSPN capabilities, including the abilities to model the full distribution of the data, to seamlessly handle latent variables, missing values and categorical data, and to efficiently perform marginal and conditional inference. Our methodology is complemented by a novel variant of the expectation-maximization algorithm for RSPNs, enabling efficient training of our models. We evaluate our approach on a synthetic dataset as well as real-world data from the MIMIC-IV intensive care unit medical database. Our evaluation demonstrates that our approach can closely match the ground-truth data generation process on synthetic data and achieve results close to neural and probabilistic baselines while using a tractable and interpretable model.

1 Introduction

Modelling of the sequential decision-making process of physicians, as well as the patient response to different treatments, has a wide range of useful applications. Transparent and interpretable models of human decision-making behaviour have been proposed as tools for understanding, quantifying and replicating policies in clinical practice [1, 2], while the treatment response models can help to inform clinician's choices when choosing among multiple available therapies [3].

Unfortunately, the currently widely used modelling methods suffer from considerable limitations. Neural-based models are highly expressive but do not represent tractable probabilistic distributions and are thus unable to support exact joint, marginal or conditional inference, greatly limiting their versatility. This also means that such models are inherently opaque and incapable of quantifying the uncertainty of their predictions in a principled way. Meanwhile, conventional probabilistic models commonly rely on highly specialised approaches for training and inference, incur high computational costs when evaluating certain probabilistic queries and are often insufficiently expressive to faithfully represent more complex systems.

---

*Work is not related to the research done at Optum Labs.

Sum-product networks (SPNs) [4, 5], a deep learning architecture based on probabilistic circuits, largely address the above limitations, as they are highly expressive while still guaranteeing tractability. Additionally, SPNs are capable of performing fast inference on high treewidth dependencies [6], which is considered impossible for other classes of probabilistic models such as Bayesian networks [7]. Due to these appealing properties, SPNs have been successfully applied in a variety of settings, including face image completion [4], robot navigation [8], image segmentation [9], speech modelling [10] or activity recognition in videos [11]. SPNs have also been generalized to time-series data of variable length, giving rise to recurrent sum-product networks (RSPNs) [12].

In this work, we introduce a probabilistic deep generative approach, Treatment-RSPN, that leverages RSPNs for joint modelling of treatment decision-making and the dynamics of the patient's response to different treatments. As part of our framework, we develop a method for transforming conventional probabilistic graphical models (PGMs), such as dynamic Bayesian networks (DBNs) [13], into RSPNs, allowing us to bootstrap our models with a structure informed by domain knowledge and the specific task. Initializing the models in that way also benefits interpretability, as the random variables from the underlying PGM and the dependencies between them are faithfully reflected in the resulting RSPN. In order to train our models on data, we introduce a variant of the expectation-maximization (EM) algorithm optimizing the parameters of the RSPN and optionally updating its structure to more closely represent the considered distribution.

Treatment-RSPN provides *interpretable* and *tractable* models for sequential treatment regimes in partially observed settings. Additionally, these models benefit from the full range of the appealing properties of SPNs, including the existence of generic algorithms for fast and efficient evaluation of joint, marginal and conditional probabilistic queries, seamless handling of latent variables and a high degree of extensibility offered by structure learning algorithms adapting the network to better match the data. Our code will be made available at `https://github.com/ML-Health/treatment-rspn`.

2 Background

The models developed using the Treatment-RSPN framework are based on recurrent sum-product networks (RSPNs) [12], a generalization of SPNs specifically adapted for modelling time-series data. Each RSPN is composed of a top network, a template network and a bottom network. The template network serves as a generic interconnecting component representing the variables in the intermediate slices of the considered temporal sequences, while the top and bottom networks enable the special-case handling of the variables in the first and last time slices, respectively. The three networks can be stacked together to form a regular, unrolled SPN capable of representing sequences up to a certain finite length L. In this process, the top network of the RSPN is taken as the root component, the template network is repeated $L - 2$ times and interconnected with other components via dedicated placeholder nodes, and the bottom network is added as the terminating component with no further outgoing connections.

The unrolled SPN is a rooted directed acyclic graph with leaf nodes representing base distributions over random variables (such as Categorical or Gaussian) and internal nodes representing sums and products of probability functions of their children. Each edge leading from a sum node to a child node is associated with a non-negative parameter indicating the weight of the corresponding child. The weights of the edges originating at a single sum node must be normalized. A key component of the SPN inference and training methods is the SPN evaluation. Given a (partial or full) assignment to the variables in the scope of an SPN $\mathbf{x}$, the value $S(\mathbf{x})$ of an SPN can be computed as follows:

$$S(\mathbf{x}) = \begin{cases} P_S(X = \mathbf{x}) & \text{if } S \text{ is a leaf} \\ \sum_{C_i \in children(S)} w_i C_i(\mathbf{x}) & \text{if } S \text{ is a sum} \\ \prod_{C_i \in children(S)} C_i((x)) & \text{if } S \text{ is a product} \end{cases}$$

The value $S(\mathbf{x})$ corresponds to the probability of the assignment $\mathbf{x}$ under the distribution modelled by the given SPN [4].

The possibility of evaluating an SPN on a partial variables assignment provides a straightforward way for performing marginal inference, as the variables not included in an assignment are being effectively marginalized over during the computation of the probability of the assignment. This can be used for handling latent variables and missing values during both training and inference, as well as an efficient evaluation of conditional queries, leveraging the fact that $P(\mathbf{q}|\mathbf{e}) = \frac{P(\mathbf{q},\mathbf{e})}{P(\mathbf{e})}$.

3 Treatment-RSPN

3.1 Treatment-RSPN Initialization

The first step of developing a model within our framework is the initialization of its structure. This step is of particular importance for RSPNs, as their structure directly affects the assumed dependencies between the variables in the scope of the model. In general, there are multiple ways in which the structure can be initialized, including learning the structure directly from the data using one of the structure learning algorithms (such as local search [12] or oSLRAU [14]), crafting the structure manually or basing it on a known probabilistic model.

In this work, we opt for the last approach, though we may have chosen one of the other options without changing the rest of our methodology. Specifically, we base the initial structure of our RSPN models on an input-output hidden Markov model (IOHMM) [15, 16]. IOHMMs model sequences composed of inputs $u_1, ..., u_n$, unobserved (latent) states $z_1, ..., z_n$ and outputs $x_1, ..., x_n$. This makes them highly suited for modelling sequential treatment regimes, as the treatment actions at each time step can be considered as inputs, while the patient physiological variables or test results can be considered as outputs. Additionally, the dependencies between the variables in an IOHMM model are rather simple and intuitive, leading to a more interpretable model.

In order to transform an IOHMM into an RSPN structure, we first consider an unrolled Bayesian network (BN) representation of the model over a fixed number of time slices. We then convert this network into an SPN structure, gradually constructing layers for the individual variables.

The construction procedure processes the variables in the BN representation of the model in topological order, maintaining metadata about their dependent child variables. When processing each variable, our procedure first constructs an SPN sum layer with one node for each possible assignment of this variable as well as variables whose children have not yet been fully processed. This sum layer is then extended with an additional structure, which may either be composed of variable indicators or product nodes connected to such indicators. The product nodes are used in cases in which the currently processed variable has children in the network and facilitate conditioning on this variable during the SPN evaluation. The weights on the edges between the sum nodes and product nodes are determined by the conditional probability distributions modelled by the Bayesian network. We give a detailed algorithmic description of this step of the initialization in the appendix B.

After the initial conversion, the repeated portions of the resulting SPN structure modelling the variables in the intermediate time slices can be straightforwardly extracted into a template network, while the structures for the first and the last time slices can be turned into the top and bottom networks, respectively. This results in the initial form of the RSPN. Note that our transformation is fully general and could be applied to an arbitrary model as long as it is expressible as a (dynamic) Bayesian network. We provide an illustrative example of an IOHMM and its associated RSPN representation in appendix D.

3.2 Treatment-RSPN Training

After deriving the initial structure for the RSPN model, we need to learn its parameters based on data. We opt to use an adapted version of the expectation-maximization (EM) algorithm. Compared to other formulations in the literature [4, 17], our variant of the EM algorithm does not require the use of differentiation and is straightforwardly applicable to RSPNs in addition to regular SPNs. Additionally, our approach is capable of adjusting the structure of the learned RSPN so that it more closely matches the distribution of the training data.

Our EM algorithm first unrolls the RSPN for a fixed number of time steps required to fit the given training data, internally assigning identical identifiers to the nodes repeated in the unrolled SPN. This ensures that the parameters of these nodes are updated jointly and do not diverge during the training process. The algorithm then alternates between the expectation and maximization steps. In an E-step, the procedure computes the likelihood of each data point "reaching" each node, i.e. the probability of this point having been generated by the node. In an M-step, the algorithm updates the weights on the edges leading from the sum nodes as well as the internal parameters of the leaves according to these likelihoods. Optionally, the algorithm can also replace the SPN leaves with a more complex learned structure by clustering the data points weighted by their likelihoods at the given leaf and creating a

Model	Log-likelihood (↑)
	Mean ± SD
Reference IOHMM	-18842.82 ± 131.44
Treatment-RSPN	-18885.07 ± 129.20

Table 1: Log-likelihood of the synthetic test data under the reference IOHMM model used for their generation and the learned Treatment-RSPN model

Model	AUROC (↑)	F1 macro (↑)	Brier score (↓)
	Mean ± SD	Mean ± SD	Mean ± SD
Interpole	0.88 ± 0.020	0.87 ± 0.023	0.042 ± 0.021
PO-MB-IL	0.58 ± 0.088	0.49 ± 0.010	0.108 ± 0.065
PO-IRL	0.57 ± 0.016	0.44 ± 0.020	0.437 ± 0.003
Off. PO-IRL	0.57 ± 0.054	0.49 ± 0.009	0.484 ± 0.003
LSTM	0.85 ± 0.030	0.85 ± 0.023	0.049 ± 0.021
Predict most common	0.50 ± 0.000	0.49 ± 0.009	0.098 ± 0.061
Treatment-RSPN	**0.91 ± 0.010**	0.86 ± 0.008	0.045 ± 0.024

Table 2: Classification and calibration performance of the evaluated models on the one-step-ahead treatment action prediction task using the MIMIC-IV sepsis dataset. The values highlighted in bold mark significantly better results at 0.05 confidence level.

new sum node with multiple leaves modelling the clusters. We give a detailed description of our EM algorithm in the appendix C.

3.3 Interpretability

We argue that our model is more interpretable compared to other (especially neural-based) methods due to its increased transparency associated with tractability as well as its basis in a human-intuitive IOHMM model with clearly defined dependencies between the involved variables. The tractability of Treatment-RSPN allows its probing with arbitrary joint, marginal or conditional queries, which may be used to answer questions about the learned probabilistic beliefs of the model.

Additionally, the latent, hidden states introduced by the underlying IOHMM model can also be used to produce explanations of the model behaviour. For example, it is possible to inspect the latent states most commonly associated with certain predictions and to visualize their associated emission distributions, which can provide insight about typical clinical states of patients for which the model suggests a particular treatment. We give examples and more details on the interpretability of our approach in the appendix F.

4 Experiments

We evaluate the performance of the Treatment-RSPN models on a synthetically generated dataset as well as real-world data from the MIMIC-IV intensive care unit medical database [18, 19].

In the synthetic data experiment, we test whether the Treatment-RSPN can closely match the parameters of a ground truth data generation process based on an IOHMM with two inputs, two states and four possible categorical observations. As shown in table 1, the log-likelihood of the Treatment-RSPN differs from the log-likelihood of the ground-truth model by less than 0.25%, indicating a near-perfect match.

In the first MIMIC-IV experiment, we apply Treatment-RSPN to one-step-ahead treatment action prediction of the administration of vasopressors, with discretized minimum blood pressure and the total volume of intravenous fluids received by the patient serving as covariates. The task is set up as a binary classification problem, with the target values indicating whether any vasopressors were

Model	HR RMSE ($\downarrow$)	MBP RMSE ($\downarrow$)	RR RMSE ($\downarrow$)	AVG RMSE ($\downarrow$)
	Mean $\pm$ SD	Mean $\pm$ SD	Mean $\pm$ SD	Mean $\pm$ SD
LSTM	10.81 ± 0.525	14.57 ± 0.938	5.35 ± 0.137	10.24 ± 0.487
Treatment-RSPN	$\mathbf{9.54 \pm 0.267}$	$\mathbf{13.09 \pm 0.453}$	$\mathbf{4.46 \pm 0.209}$	$\mathbf{9.03 \pm 0.286}$

Table 3: Regression performance of the evaluated models on the one-step ahead treatment response prediction task using the MIMIC-IV sepsis dataset. The values highlighted in bold mark significantly better results at 0.05 confidence level.

administered at the corresponding time step. We compare the performance of our model to baselines from [1], as well as a basic model always predicting the most common treatment action. The results of the experiment are captured in table 2. The Treatment-RSPN achieved a statistically significantly improved result over all baselines in the AUROC score and is on par with Interpole [1] (the best performing baseline) in terms of F1 macro and Brier score.

In the second MIMIC-IV experiment, we evaluate the performance of Treatment-RSPN on one-step-ahead continuous treatment response prediction, using vasopressors and intravenous fluids administration indicators as inputs and heart rate, minimum blood pressure and respiratory rate as the predicted treatment outcome variables. This task can thus be seen as a regression task with three predicted variables. The results of Treatment-RSPN and an LSTM baseline [20] on this task are shown in table 3. We can see that the Treatment-RSPN model significantly outperforms the LSTM baseline.

In both MIMIC-IV experiments, the Treatment-RSPN achieved a performance competitive with the baselines, demonstrating the promise of the method.

5 Conclusion and Future Work

In this paper, we introduced a framework for tractable and interpretable modelling of clinical decision-making and treatment response mechanisms using recurrent-sum-product networks. We discussed the process for constructing the initial network structure for the RSPN models, as well as an adapted version of the EM algorithm suitable for their training. In the future, we would like to extend the structure-learning capabilities of our training algorithm and further explore how the interpretability and tractability of our models could be used to provide more informative predictions. We would also like to explore the potential use of Treatment-RSPN for counterfactual prediction to estimate treatment effects under a dynamic treatment regime [21].

Acknowledgments and Disclosure of Funding

References

[1] Alihan Hüyük, Daniel Jarrett, Cem Tekin, and Mihaela Van Der Schaar. Explaining by imitating: Understanding decisions by interpretable policy learning. In *International Conference on Learning Representations*, 2020.

[2] Alizée Pace, Alex Chan, and Mihaela van der Schaar. POETREE: Interpretable policy learning with adaptive decision trees. In *International Conference on Learning Representations*, 2022.

[3] Bryan Lim. Forecasting treatment responses over time using recurrent marginal structural networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[4] Hoifung Poon and Pedro Domingos. Sum-product networks: A new deep architecture. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 689–690. IEEE, 2011.

[5] Adnan Darwiche. A differential approach to inference in bayesian networks. *Journal of the ACM (JACM)*, 50(3):280–305, 2003.

[6] Antonio Vergari, Nicola Di Mauro, and Floriana Esposito. Simplifying, regularizing and strengthening sum-product network structure learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 343–358. Springer, 2015.

[7] Johan H. P. Kwisthout, Hans L. Bodlaender, and L. C. van der Gaag. The necessity of bounded treewidth for efficient inference in bayesian networks. In *Proceedings of the 2010 Conference on ECAI 2010: 19th European Conference on Artificial Intelligence*, page 237–242, NLD, 2010. IOS Press.

[8] Kaiyu Zheng, Andrzej Pronobis, and Rajesh Rao. Learning graph-structured sum-product networks for probabilistic semantic maps. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[9] Abram L Friesen and Pedro Domingos. Submodular sum-product networks for scene understanding, 2016.

[10] Robert Peharz, Georg Kapeller, Pejman Mowlaee, and Franz Pernkopf. Modeling speech with sum-product networks: Application to bandwidth extension. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3699–3703, 2014.

[11] Mohamed R Amer and Sinisa Todorovic. Sum product networks for activity recognition. *IEEE transactions on pattern analysis and machine intelligence*, 38(4):800–813, 2015.

[12] Mazen Melibari, Pascal Poupart, Prashant Doshi, and George Trimponias. Dynamic sum product networks for tractable inference on sequence data. In *Conference on Probabilistic Graphical Models*, pages 345–355. PMLR, 2016.

[13] Thomas Dean and Keiji Kanazawa. A model for reasoning about persistence and causation. *Computational intelligence*, 5(2):142–150, 1989.

[14] Agastya Kalra, Abdullah Rashwan, Wei-Shou Hsu, Pascal Poupart, Prashant Doshi, and Georgios Trimponias. Online structure learning for feed-forward and recurrent sum-product networks. *Advances in Neural Information Processing Systems*, 31, 2018.

[15] Yoshua Bengio and Paolo Frasconi. An input output hmm architecture. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7. MIT Press, 1994.

[16] Y. Bengio and P. Frasconi. Input-output hmms for sequence processing. *IEEE Transactions on Neural Networks*, 7(5):1231–1249, 1996.

[17] Robert Peharz, Robert Gens, Franz Pernkopf, and Pedro Domingos. On the latent variable interpretation in sum-product networks. *IEEE transactions on pattern analysis and machine intelligence*, 39(10):2030–2044, 2016.

[18] Alistair Johnson, Lucas Bulgarelli, Tom Pollard, Steve Horng, Leo Anthony Celi, and Roger Mark. MIMIC-IV (version 1.0). *PhysioNet*, 2021.

[19] Ary L Goldberger, Luis AN Amaral, Leon Glass, Jeffrey M Hausdorff, Plamen Ch Ivanov, Roger G Mark, Joseph E Mietus, George B Moody, Chung-Kang Peng, and H Eugene Stanley. Physiobank, physiotoolkit, and physionet: Components of a new research resource for complex physiologic signals. *Circulation*, 101(23):e215–e220, 2000.

[20] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, nov 1997.

[21] Rui Li, Stephanie Hu, Mingyu Lu, Yuria Utusumi, Prithwish Chakraborty, Daby Sow, Piyush Madan, Mohamed Ghalwash, Zach Shahn, and Li-wei H Lehman. G-net: a recurrent network approach to g-computation for counterfactual prediction under a dynamic treatment regime. *Proceedings of Machine Learning for Health*, 2021.

[22] D Jarrett and Mihaela van der Schaar. Inverse active sensing: modeling and understanding timely decision-making. In *ICML*, 2020.

[23] T Makino and J Takeuchi. Apprenticeship learning for model parameters of partially observable environments. In *ICML*, 2012.

[24] P Englert, A Paraschos, J Peters, and M.P. Deisenroth. Probabilistic model-based imitation learning. *Adaptive Behavior*, 21:388–403, 2013.

[25] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

A Treatment-RSPN Overview Diagram

An overview diagram illustrating our approach is provided in figure 1.

B Conversion Procedure

We give our procedure for converting a Bayesian network into an SPN in algorithm 1.

Algorithm 1: BN to SPN conversion procedure

Given a Bayesian network N to convert to an SPN

1. Set Vs to be the variables (nodes) in N in the topological order.

2. Set $c = \emptyset$ to be the set of variables currently conditioned on.

3. Set $d = \{\}$ to be a map of unresolved variable dependencies (mapping from parent variables to lists of child variables).

4. For V in Vs:

   4.1. Construct a layer of sum nodes s, one for each possible assignment $\mathbf{x}_i^s$ of variables in c or a single sum node if c is empty. Attach the layer to the previous product layer p if any, with edges leading between nodes n_j^p and n_i^s whenever $\mathbf{x}_j^p$ and $\mathbf{x}_i^s$ for these nodes are compatible.

   4.2. If V has children in N:

       a. Add V to c.

       b. Remove all occurrences of V from d, remove any variables with empty child list in d from c.

       c. Add $V \mapsto \text{children}_N(V)$ to d.

       d. Add a product layer p with one node for each possible assignment $\mathbf{x}_j^p$ of variables in c, each connected to an indicator leaf for $V = v_j$ where v_j is the value of V in $\mathbf{x}_j^p$. Connect the product layer to the previous sum layer s, with edges leading between nodes n_i^s and n_j^p whenever $\mathbf{x}_i^s$ and $\mathbf{x}_j^p$ for these nodes are compatible and edge weights determined by $P_N(V = v_j | \mathbf{x}_i^s)$.

       Otherwise:

       a. Construct a layer of leaf nodes for V with distributions and connections to the previous layer l representing the distribution $P_N(V | \mathbf{x}_i^l)$ for each node n_i^l with assignment $\mathbf{x}_i^l$ in l.

C EM Algorithm

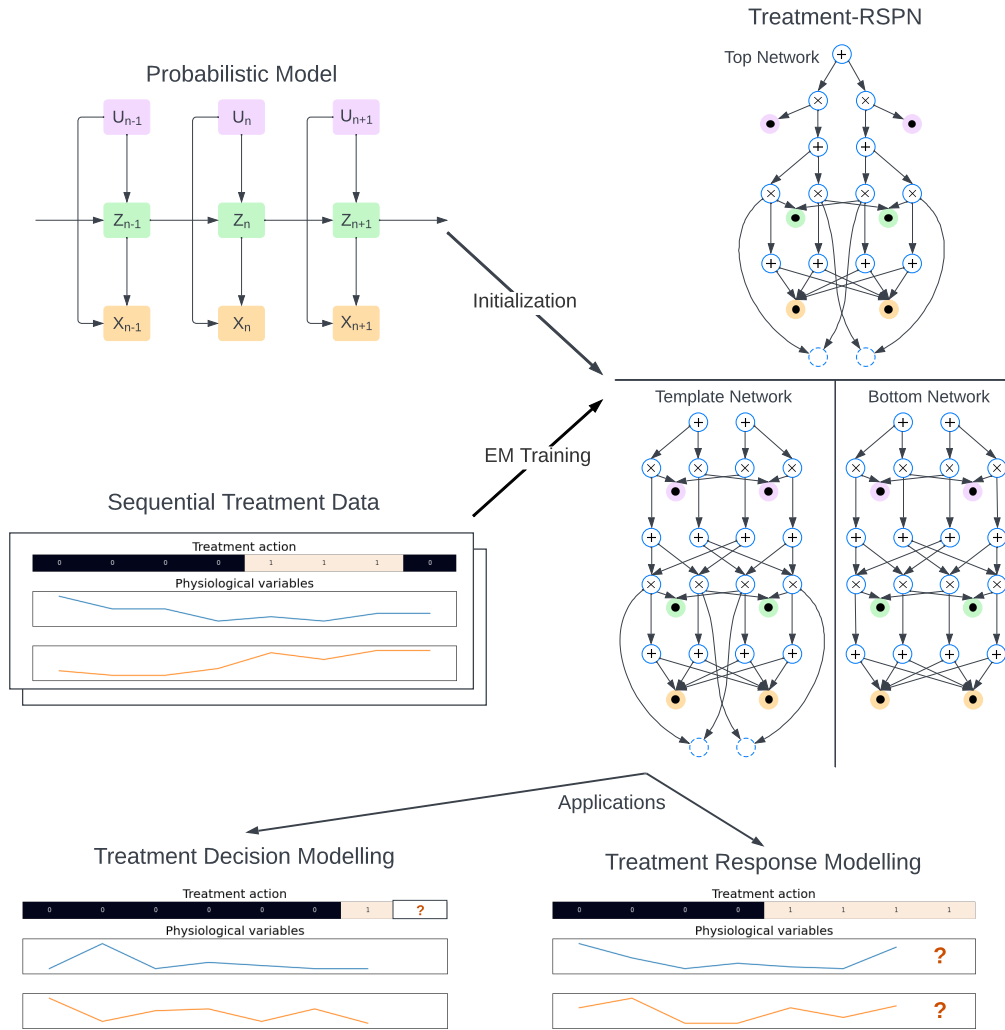We detail our variant of the RSPN expectation maximization procedure in algorithm 2.

Figure 1: Overview of the Treatment-RSPN framework. Treatment-RSPN is initialised based on a probabilistic model (e.g. IOHMM) and trained on sequential treatment data using our expectation maximization algorithm. The resulting model can be applied to different tasks, such as treatment action prediction and treatment response prediction.

Algorithm 2: RSPN expectation maximization

Given an RSPN R and the training dataset D

1. Unroll R into a regular SPN S matching the length of the sequences in D and using shared node identifiers for repeated nodes.

2. Evaluate S on D using procedure described in appendix 2, store the value computed at each node.

3. Set $pl = \{(\text{nil}, \text{root}(R)) \mapsto \mathbf{1}\}$ to be the map of data point likelihoods passed to each node by its parents. The root node has no parents, and is thus passed likelihood of 1 for each point.

4. Set $l = \{(\text{id}(p), \text{id}(c)) \mapsto 0\}$ for all pairs of parents and children (p, c) in S. l stores the aggregated likelihoods of data points reaching nodes from their parents.

5. Set $lp, lw = \{\}, \{\}$ to store data points and their likelihoods at each leaf.

6. For n in nodes(S), processed from top to bottom:

    6.1. Set $dl = \sum_{p \in \text{parents}(n)} pl[p, n]$ to be the likelihoods of reaching the current node for each data point, passed to n by its parents.

    6.2. Increment $l[\text{id}(p), \text{id}(n)]$ by $\sum_{d \in D} pl[p, n, d]$ for all p in parents(n).

    6.3. If n is a sum:

        a. Set $cl = \{c \mapsto \text{weight}(n, c) \times \text{value}(c)\}$ for all c in children(n) to store the base likelihood of each child.

        b. Normalize the likelihoods of the children in cl

        c. Set $pl[n, c] = cl[c] \times dl$

    If n is a product:

        a. Set $pl[n, c] = dl$ for all c in children(n)

    If n is a leaf node:

        a. Store the current data D and their likelihoods at this node dl in $lp[n]$ and $lw[n]$.

7. Update the weights of each edge between a sum node s and its child node c to $\frac{l[\text{id}(s), \text{id}(c)]}{\sum_{c' \in \text{children}(s)} l[\text{id}(s), \text{id}(c')]}$

8. Update the distributions in leaves according to the weighted data in lp and lw or cluster the weighted data and introduce a sub-structure to model the mixture.

D   IOHMM and Treatment-RSPN Diagrams

To gain more intuition about the transformation process and the structure of the resulting Treatment-RSPN, consider the example IOHMM in figure 2 and the corresponding RSPN obtained using our conversion procedure in figure 3. There are several key observations to note. First, notice how the product nodes are naturally used to model conditioning on a variable represented by leaves attached to them. This is exploited by the transformation algorithm, which constructs a product layer for each variable with children in the underlying BN. Second, note how the converging connections from multiple product nodes effectively "cancel" conditioning on a variable. For example, the edges from the product nodes for the assignments $u_1 = 0, z_1 = 0$ and $u_1 = 1, z_1 = 0$ leading into a single node stop the conditioning on the variable U_1. Finally, note that the probabilities from the conditional probability tables associated with the IOHMM model are repeated several times in the network. During the training of our model, we assign identical parameters to the nodes surrounding the edges with these probabilities, which ensures that they are updated in unison.

E   Experiment Baselines

Treatment Action Prediction. We compare our model with the baselines from [1]. We utilize Interpole [1], which is a Bayesian method for interpretable treatment policy learning, PO-IRL, which is based on inverse reinforcement learning [22], Off. PO-IRL, which is an offline variant of PO-IRL [23], PO-MB-IL, which performs model-based imitation learning [24], and an LSTM, a gated recurrent network. The PO-IRL and Off. PO-IRL are initialized using the conventional algorithms for learning an IOHMM and further trained using Markov chain Monte Carlo sampling, while PO-MB-IL is trained using an EM algorithm after initialization. The LSTM is trained using Adam optimizer [25] with learning rate of 0.001. Further details about these baselines can be found in the original paper [1]. All the baselines are trained with their default parameters. Additionally, we also report

$P(U_n = 0)$	$P(U_n = 1)$
0.5	0.5

U_1	$P(Z_1 = 0 \mid U_1)$	$P(Z_1 = 1 \mid U_1)$
0	0.6	0.4
1	0.3	0.7

Z_{n-1}	U_n	$P(Z_n = 0 \mid Z_{n-1}, U_n)$	$P(Z_n = 1 \mid Z_{n-1}, U_n)$
0	0	0.8	0.2
0	1	0.25	0.75
1	0	0.65	0.35
1	1	0.1	0.9

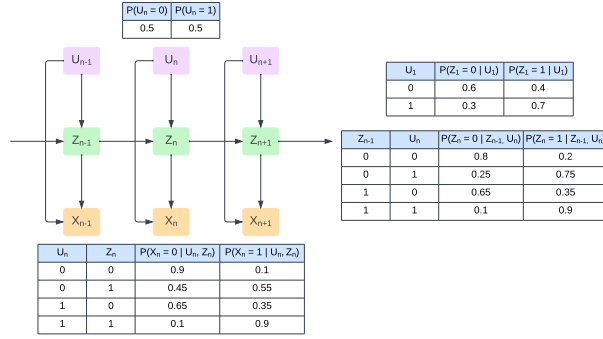| U_n	Z_n	$P(X_n = 0 \mid U_n, Z_n)$	$P(X_n = 1 \mid U_n, Z_n)$
0	0	0.9	0.1
0	1	0.45	0.55
1	0	0.65	0.35
1	1	0.1	0.9

Figure 2: A simple IOHMM with discrete variables for the inputs, latent states and observations.
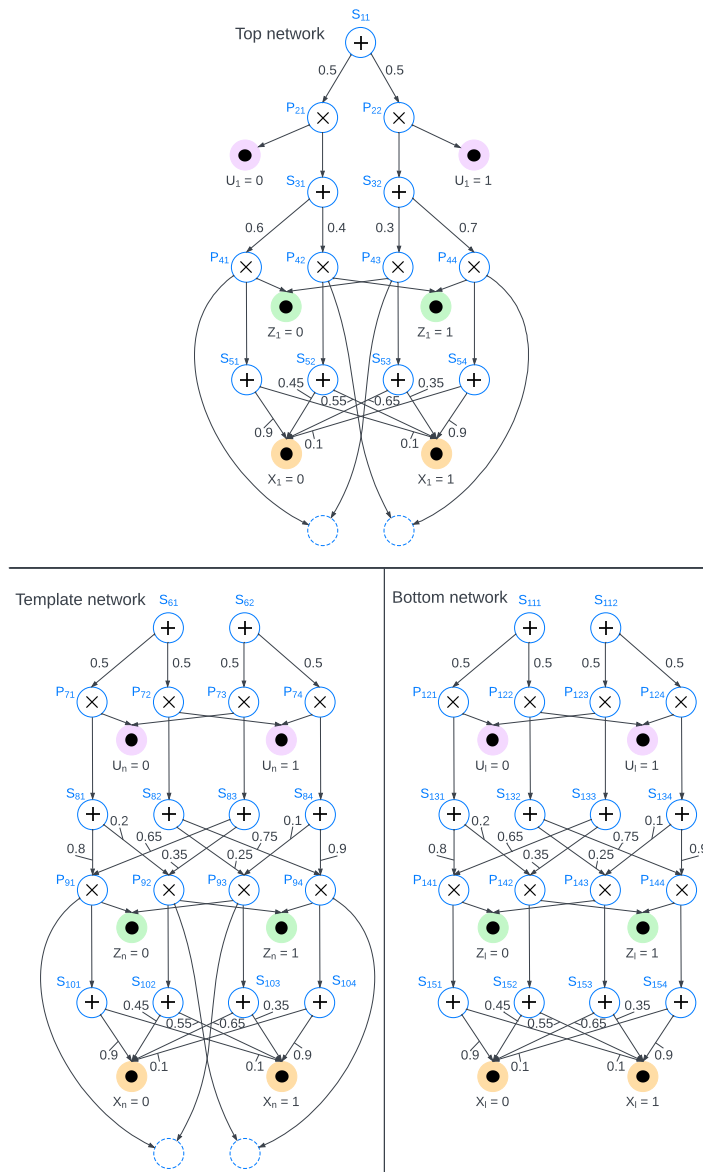

Figure 3: An RSPN corresponding to the IOHMM model shown in figure 2. This model can be obtained by running our initialization procedure described in section 3.1.

(a) States for negative vasopressor treatment
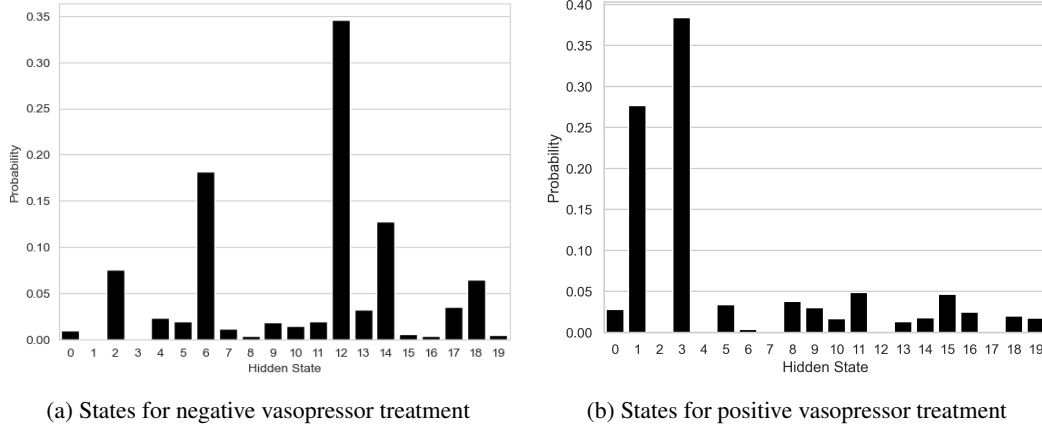
(b) States for positive vasopressor treatment

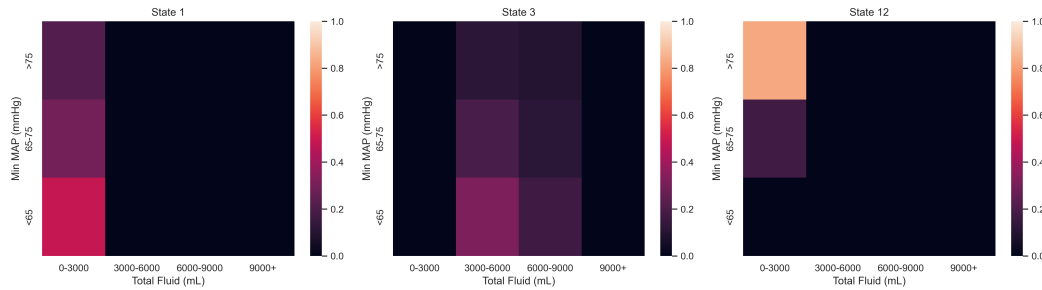Figure 4: States with the highest likelihoods before a negative/positive vasopressor treatment

Figure 5: Emission distributions for selected states with the highest probabilities before positive vasopressor treatment (states 1 and 3) and negative vasopressor treatment (state 12)

the valuation results for a simple classifier which always predicts the most likely treatment (with the probability equal to the portion of time steps with this treatment).

Treatment Response Prediction. We compare our model with an LSTM [20]. The network is composed of an LSTM unit with a hidden size of 32 and a linear output layer. We train the network for 30 epochs using the Adam optimizer (running further training iterations did not have any significant effect on the results).

F Interpretability

In section 3.3 of the main text, we argued that Treatment-RSPN is more interpretable compared to the other approaches due to its probabilistic nature, tractability and basis in a known probabilistic graphical model with intuitive dependencies between the variables. Here, we show an illustrative example of how these properties could be practically utilized to gain insight into the learned beliefs of the model. In our analysis, we focus on the Treatment-RSPN model used for the treatment action prediction experiment.

As a first step, we determine the most likely states to occur immediately before the positive or negative vasopressor treatment action. This is done by computing the likelihoods of the different states at each time step of each temporal sequence in the test set and aggregating them appropriately. The results are visualized in figure 4, showing that the most likely state before a negative vasopressor treatment is state number 12, while the most likely states before a positive vasopressor treatment are states number 1 and 3.

In order to gain further insight into the characteristics of these states, we can examine the emission distributions associated with them. These are visualized in figure 5. Each heatmap depicts the probabilities for the observed outcomes in the current time step based on the hidden state. Both states 1 and 3 occur at timesteps where the observed blood pressure is low. This corresponds with the

expectations, as clinicians are likely to begin vasopressor therapy for urgently hypotensive patients. It thus seems that the beliefs of the model are clinically meaningful, as it has learned that vasopressors must be administered after low blood pressure. In comparison, patients in state 12, which is predictive of negative vasopressor treatment, are much more likely to have higher blood pressure, indicating that the therapy is not needed.

In addition to the above observations, we can also note that the model apparently learned to distinguish between patients who have been administered a larger volume of intravenous fluids (state 3) and patients who have not received much of these fluids (state 1). This may potentially be helpful for determining the point at which the vasopressor treatment is more likely to cease.

G   Discrete Treatment Response Prediction Experiment

In addition to the experiments described in section 4, we also evaluated our model on the "mirror" task to MIMIC-IV experiment one, in which the model performs a one-step-ahead prediction of the discretized covariates (minimum blood pressure and the total volume of intravenous fluids). The results for this task are shown in table 4.

Model	F1 weighted ($\uparrow$)	AUROC weighted ($\uparrow$)	Brier score ($\downarrow$)
	Mean $\pm$ SD	Mean $\pm$ SD	Mean $\pm$ SD
LSTM	0.64 ± 0.054	0.93 ± 0.012	0.49 ± 0.058
Predict most common	0.10 ± 0.062	0.50 ± 0.000	0.87 ± 0.031
Treatment-RSPN	0.60 ± 0.059	0.93 ± 0.011	0.52 ± 0.062

Table 4: Classification performance of the evaluated models on the one-step ahead treatment response prediction task using the MIMIC-IV sepsis dataset