Tensor networks for active inference with discrete observation spaces

Samuel T. Wauthier[1], **Bram Vanhecke**[2], **Tim Verbelen**[1], **Bart Dhoedt**[1]
[1]IDLab, Ghent University – imec
{first name}.{last name}@ugent.be
[2]University of Vienna
bram.andre.roland.vanhecke@univie.ac.at

Abstract

In recent years, quantum physics-inspired tensor networks have seen an explosion in use cases. While these networks were originally developed to model many-body quantum systems, their usage has expanded into the field of machine learning, where they are often used as an alternative to neural networks. In a similar way, the neuroscience-based theory of active inference, a general framework for behavior and learning in autonomous agents, has started branching out into machine learning. Since every aspect of an active inference model, such as the latent space structure, must be manually defined, efforts have been made to learn state space representations automatically from observations using deep neural networks. In this work, we show that tensor networks can be employed to learn an active inference model with a discrete observation space. We demonstrate our method on the T-maze problem and show that the agent acts Bayes optimal as expected under active inference.

1 Introduction

Active inference comprises a general framework for behavior and learning in autonomous agents [3]. It starts from the Bayesian brain hypothesis, which assumes that the brain functions as a Bayesian statistician (constantly updating its beliefs about the environment using Bayes' rule), and combines perception and action into a single functional which serves as the fundamental quantity which all autonomous agents attempt to minimize. The latter is referred to as the variational free energy. In short, an autonomous agent will update its beliefs about the environment and select actions in order to minimize its variational free energy.

Crucially, an active inference model contains a generative model, which embodies the agent's beliefs about the environment. Traditionally, the latent space structure, transition dynamics and likelihood of the model must be specified by hand. Deep active inference attempts to remedy this problem by learning these aspects of the model from the data [9, 1].

Tensor networks, which were originally developed to model many-body quantum states, are architectures of contractions between tensors. In recent years, tensor networks have been used in other areas, such as supervised learning [7], unsupervised learning [5, 2, 10] and natural language processing [8]. Moreover, they have proven to be powerful tools for generative modeling.

In this paper, we show that a matrix product state, which is a particular tensor network architecture, can be used to build a generative model of sequential data when observations are discrete. Moreover, we show that such a model can subsequently be used to perform action selection with the active inference scheme. We demonstrate the method on the T-maze problem.

Section 2 provides a brief account of active inference, explains how tensor networks can be used as generative model, and describes the T-maze environment. In section 3, we present and discuss the

Machine Learning and the Physical Sciences workshop, NeurIPS 2022.

results obtained on the T-maze environment. Section 4, summarizes our findings and concludes this work.

2 Methods

Active inference An active inference agent's generative model can be formalized as a partially observable Markov decision process (POMDP) with the following probability distribution [3]:

$$P(\tilde{o}, \tilde{s}, \pi) = P(s_0)P(\pi)\prod_{t=1}^{T} P(o_t|s_t)P(s_t|s_{t-1}, \pi), \tag{1}$$

where o_t and s_t are observation and latent state at time step t, $\pi(t) = a_t$ is a policy which provides an action at each time step and we have introduced the tilde notation to indicate a sequence over time $\tilde{x} = (x_1, x_2, \ldots, x_T)$. The agent will attempt to minimize its variational free energy, defined as [3]

$$\begin{aligned} F &= \mathrm{E}_Q\left[\log Q(\tilde{s}, \pi) - \log P(\tilde{o}, \tilde{s}, \pi)\right] \\ &= \underbrace{\mathrm{D}_{\mathrm{KL}}(Q(\tilde{s}, \pi) \,||\, P(\tilde{s}, \pi))}_{\text{complexity}} - \underbrace{\mathrm{E}_Q\left[\log P(\tilde{o}|\tilde{s}, \pi)\right]}_{\text{accuracy}} \end{aligned} \tag{2}$$

with Q an approximate posterior distribution. More importantly, the agent will select actions in such a way that it will continue to minimize free energy in the future. In other words, it will minimize its expected free energy. In the sophisticated active inference scheme [4], expected free energy is given by

$$\begin{aligned} G(o_\tau, a_\tau) = &\underbrace{\mathrm{D}_{\mathrm{KL}}(Q(o_{\tau+1}|a_{<\tau+1}) \,||\, P(o_{\tau+1})) + \mathrm{E}_{Q(s_{\tau+1}|a_{<\tau+1})}[\mathrm{H}(P(o_{\tau+1}|s_{\tau+1}))]}_{\text{expected free energy of next action}} \\ &+ \underbrace{\mathrm{E}_{Q(a_{\tau+1}|o_{\tau+1})Q(o_{\tau+1}|a_{<\tau+1})}[G(o_{\tau+1}, a_{\tau+1})]}_{\text{expected free energy of subsequent actions}} \end{aligned} \tag{3}$$

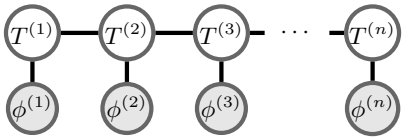
$$Q(a_\tau|o_\tau) = \sigma[-G(o_\tau, a_\tau)], \tag{4}$$

where $P(o_{\tau+1})$ corresponds to the preferred state of the agent, i.e. the state the agent wants to be in.

Generative modeling with tensor networks Similarly to Han et al. [5], using the Born rule,

$$P(X = x) = \frac{|\Psi(x)|^2}{Z}, \tag{5}$$

with $Z = \sum_{\{x\}} |\Psi(x)|^2$ and $x = (x_1, x_2, \ldots, x_n)$, the generative model in Eq. (1) can be represented by a quantum wave function $\Psi(x)$. In turn, this wave function may be parameterized by a tensor network. One of the simplest forms of tensor networks is the matrix product state (MPS), a.k.a. tensor train. This entails a parameterization of the form

$\Psi(x) =$ 

where we have used graphical notation, such that each node corresponds to a tensor and each edge indicates a tensor contraction, $T^{(i)}$ are weight tensors and $\phi^{(i)} = \phi^{(i)}(x_i)$ are feature maps which map each value x_i onto a vector.

Special attention must be paid to the feature maps $\phi^{(i)}(x_i)$, since they determine whether the model can be given the generative interpretation [7]. In particular, the components of a feature map must be orthonormal functions. Moreover, for any input x_i to define a probability distribution, the resulting vectors $\phi^{(i)}$ must have norm 1. For cases where x is discrete and can take on a finite set of values, a one-hot encoding satisfies these conditions.

An MPS can be trained using a variation of the density matrix renormalization group (DMRG) algorithm[7]. Given a data set and after defining a loss function, this algorithm updates the weight

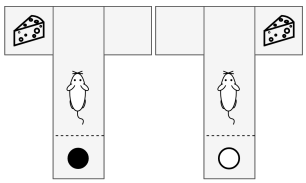


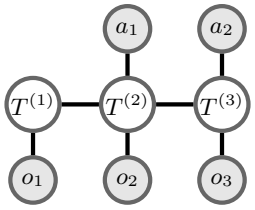
Figure 1: The T-maze problem. The mouse start in the center location and wants to find the cheese located in either the left or right branch. A full (left) or empty (right) cue indicates the cheese is in the left or right branch, respectively.

tensors by "sweeping" back and forth across the MPS. To train the generative model, we maximize the negative log-likelihood (NLL) [5], i.e. we maximize the model evidence directly. After training, the tensor network can be used to infer probability densities over unobserved variables by contracting the MPS with observed values.

T-maze environment We show the workings of the generative model using tensor networks on the T-maze problem (Fig. 1) [3, 6]. Importantly, the environment is discrete, providing discrete observations and receiving discrete actions. In this environment, an agent, such as a mouse, sits within a T-shaped maze. The mouse's goal is to obtain a reward: a piece of cheese located in one of the branches of the T. Starting out in the center position, the mouse can move to the left branch, right branch or cue location at the bottom of the T. If the mouse chooses to go to the left or right branch, it may obtain the cheese, but it cannot leave the branch and, therefore, may never obtain the cheese if it chooses wrongly. However, if the mouse goes to the cue location first, the location of the cheese will be revealed and the environment becomes deterministic. As such, the mouse has a total of two actions to solve the environment. More technically, the T-maze provides observations containing the position, reward and context (i.e. cue state). The environment can receive actions corresponding to moving to the center, right, left and cue location. In summary, the T-maze environment is used to test whether an agent will first attempt to resolve any uncertainty in the environment in order to obtain a higher probability of reaching the reward.

To accommodate for sequential data with actions and observations as defined by active inference and provided by the environment, we adapted the MPS described above. Each tensor $T^{(i)}$ now takes two inputs: action a_i and corresponding observation o_i. The network takes the form

$$\Psi(x) =$$



where we used a_i and o_i to denote the vectors obtained after applying feature maps (one-hot encodings) corresponding to action a_i and observation o_i. Since we define that action a_i leads to observation o_{i+1}, the tensor $T^{(1)}$ does not receive an action input.

3 Results and discussion

This section demonstrates how the generative model behaves when certain actions and observations are given. Moreover, it shows how the agent performs action selection under the active inference scheme.

The data used in this experiment was constructed by generating one of every possible path through the maze, i.e. 202 sequences of actions and observations. The model was trained over 500 epochs with a batch size of 10, where one epoch consisted of one right-to-left-to-right sweep per batch. The learning rate was set to 10^{-4} and was further reduced by 10 % whenever the loss increased too much (i.e. by more than 0.5). Additionally, bonds started with 8 dimensions. The singular value cutoff point was set to 10 % of the largest singular value.

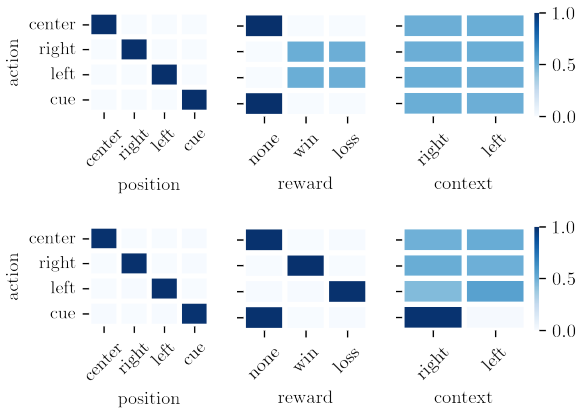


Figure 2: (top) Model beliefs over observation o_2 given action a_1 per modality. (bottom) Model beliefs for observation o_3 given action a_2 per modality, when the agent has observed cue empty.

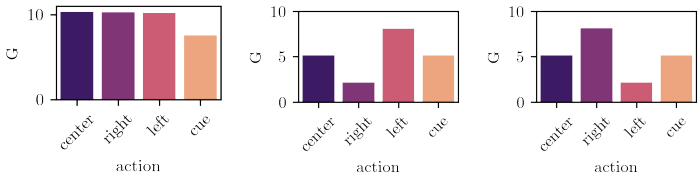


Figure 3: Expected free energy per action for (left) action a_1, and for action a_2 when observation o_2 was (center) cue full and (right) cue empty.

Beliefs Figure 2 shows how the agent's beliefs (i.e. probabilities within the generative model) shift depending on the actions it takes and which observations it receives. The top row shows the agent's beliefs for action a_1. It shows that the agent is always certain it will end up in the location it chose to go to, however, it is uncertain whether it will receive a reward if it decides to go to the left or right branch. Moreover, it doesn't know whether the cue is full or empty. The bottom row shows the agent's beliefs after having taken action 3 and observed an empty cue. The agent is still certain of where it will end up, but now it also certain it will obtain a reward if it goes to the right branch, as well as no reward if it goes to the left branch. In addition, it is certain it will observe an empty cue if it decides to stay in the cue location.

Action selection Figure 3 shows the expected free energy (Eq. (3)) per action using preferences

$$P(\text{position}) = \sigma([0 \quad 0 \quad 0 \quad 0]), \quad P(\text{reward}) = \sigma([0 \quad 3 \quad -3]), \quad P(\text{context}) = \sigma([0 \quad 0]). \quad (6)$$

The leftmost panel shows the expected free energy for action a_1. Here, going to the cue gives the lowest expected free energy. This is according to our expectations, since this action will reduce uncertainty in the environment and make sure the agent knows where the reward is located. The central and rightmost panels show the expected free energy for action a_2 when the agent has gone to the cue and observed cue empty and cue full, respectively. Now, the agent is certain about the reward's location, which is reflected by a lower expected free energy, and will go right or left depending on the cue. As such, the agent takes the Bayes optimal path towards the reward.

4 Conclusion

We introduced an active inference model based on tensor networks that is able to learn from sequential data. We showed how the beliefs of an agent employing such a model shift when given observations and how its actions are chosen based on the expected free energy per action.

In the future, we plan to apply tensor network-based active inference agents to other environments, as well as make an in-depth comparison with neural networks. This will better establish the benefits and drawbacks of the method, as well as, allow broadening of the range of applications.

Impact statement

A lot of focus is placed on quantum computers nowadays, as they are expected to outperform classical computers on many computational problems in the near future. Tensor networks are especially important, since they can be directly mapped to quantum circuits. Therefore, they have the potential to have a huge impact on future machine learning models in the transition towards quantum machine learning.

Acknowledgments and Disclosure of Funding

This research received funding from the Flemish Government under the "Onderzoeksprogramma Artificiële Intelligentie (AI) Vlaanderen" programme. This work has received support from the European Union's Horizon 2020 program through Grant No. 863476 (ERC-CoG SEQUAM).

References

[1] Ozan Çatal, Samuel Wauthier, Cedric De Boom, Tim Verbelen, and Bart Dhoedt. Learning generative state space models for active inference. *Frontiers in Computational Neuroscience*, 14:103, 2020. ISSN 1662-5188. doi: 10.3389/fncom.2020.574372.

[2] Song Cheng, Lei Wang, Tao Xiang, and Pan Zhang. Tree tensor networks for generative modeling. *Phys. Rev. B*, 99:155131, Apr 2019. doi: 10.1103/PhysRevB.99.155131.

[3] Karl Friston, Thomas FitzGerald, Francesco Rigoli, Philipp Schwartenbeck, John O'Doherty, and Giovanni Pezzulo. Active inference and learning. *Neuroscience & Biobehavioral Reviews*, 68:862–879, 2016. ISSN 0149-7634. doi: 10.1016/j.neubiorev.2016.06.022.

[4] Karl Friston, Lancelot Da Costa, Danijar Hafner, Casper Hesp, and Thomas Parr. Sophisticated Inference. *Neural Computation*, 33(3):713–763, Mar 2021. ISSN 0899-7667. doi: 10.1162/neco_a_01351.

[5] Zhao-Yu Han, Jun Wang, Heng Fan, Lei Wang, and Pan Zhang. Unsupervised generative modeling using matrix product states. *Phys. Rev. X*, 8:031012, Jul 2018. doi: 10.1103/PhysRevX.8.031012.

[6] Conor Heins, Beren Millidge, Daphne Demekas, Brennan Klein, Karl Friston, Iain D. Couzin, and Alexander Tschantz. pymdp: A python library for active inference in discrete state spaces. *Journal of Open Source Software*, 7(73):4098, 2022. doi: 10.21105/joss.04098.

[7] Edwin Stoudenmire and David J Schwab. Supervised learning with tensor networks. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[8] Jirawat Tangpanitanon, Chanatip Mangkang, Pradeep Bhadola, Yuichiro Minato, Dimitris G Angelakis, and Thiparat Chotibut. Explainable natural language processing with matrix product states. *New Journal of Physics*, 24(5):053032, May 2022. doi: 10.1088/1367-2630/ac6232.

[9] Kai Ueltzhöffer. Deep active inference. *Biological Cybernetics*, 112(6):547–573, Dec 2018. ISSN 1432-0770. doi: 10.1007/s00422-018-0785-7.

[10] Tom Vieijra, Laurens Vanderstraeten, and Frank Verstraete. Generative modeling with projected entangled-pair states, 2022.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [No] Page limitations did not allow for further elaboration.

(c) Did you discuss any potential negative societal impacts of your work? [N/A]

(d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...

   (a) Did you state the full set of assumptions of all theoretical results? [N/A]

   (b) Did you include complete proofs of all theoretical results? [N/A]

3. If you ran experiments...

   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No] This will be released with future work.

   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]

   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]

   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [No] It was all run locally on a laptop.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

   (a) If your work uses existing assets, did you cite the creators? [Yes]

   (b) Did you mention the license of the assets? [N/A]

   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]

   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]