THESIS PROPOSAL: HIDDEN MARKOV MODELS FOR MULTI-VIEW REINFORCEMENT LEARNING WITH FAULTY SENSORS

Tom Lieberum, Supervision by David Kuric

1 RESEARCH QUESTION AND LITERATURE REVIEW

While there already exist multiple approaches to multi-view deep learning in general (Zhao et al., 2017; Wang et al., 2019; Qin et al., 2019; Federici et al., 2020) and multi-view reinforcement in particular (Li et al., 2019; Fan & Li, 2021; Fadadu et al., 2021), we found the existing literature to be lacking along the following dimensions:

- Restricting the mutual information $I(x_t^1, x_t^2)$ between views 1 and 2. An important subcategory of this restriction is the "mutual redundancy" assumption that every information which is not shared is irrelevant. [1]
- Assuming a mean field approximation over views, i.e. assuming that the latent variable z can be partitioned into z^1 and z^2, such that $p(z_t|x_t^1, x_t^2) = p(z_t^1|x_t^1)p(z_t^2|x_t^2)$.
- Providing no principled way to deal with unobserved views, e.g. in the case of malfunctioning sensors at test time.[2]

In recent years, model-based reinforcement learning has seen significant progress (Ha & Schmidhuber, 2018; Hafner et al., 2019a;b; Schrittwieser et al., 2020; Hafner et al., 2020; Ye et al., 2021; Ozair et al., 2021). In particular, Hafner et al. (2020); Ozair et al. (2021) demonstrated the efficacy of discrete world models, i.e. learning a world model with discrete latent states z_t.

However, such models may encounter difficulties in a setting with multiple views, unobserved views or malfunctioning sensors. In this work, we will propose a general method for dealing with multiple views which does not make these restricting assumptions and allows us to deal in a principled and theoretically grounded manner with unobserved views. We will show that pre-existing methods are not sufficient to dealing with the aforementioned issues even when extended to accomodate multiple views.

The only other work that we are aware of that treats multi-view RL somewhat holistically (Li et al., 2019) has several shortcomings that we wish to address. First, it only considers observing one possible view at a time. And second, it suffers from the assumptions stated above.

Inspired by the results in Hafner et al. (2020), we model the RL environment as a POMDP with discrete latent variables z_t and the assumption that views are independently generated given the current latent state, i.e. $p(x_t^1, x_t^2|z_t) = p(x_t^1|z_t)p(x_t^2|z_t)$. This independence assumption is defensible under the view that the latent state of the POMDP renders it Markovian, i.e. that the world state is completely determined/captured by the latent state z_t.

As we will see in section 2.2, this discreteness modeling choice will allow us, for reasonable state space sizes, to exactly compute $p(x_t|z_t)p(z_t)$ which will be the cornerstone of the multi-view part of this project. For large state spaces, e.g. 3D environments, it seems likely to achieve at least reasonable approximations.

[1] This is a reasonable assumption if we assume that view 2 is a data augmented version of view 1, but is clearly not true in the general setting.

[2] This is less of a problem under the "mutual redundancy" assumption mentioned before.

2 METHODOLOGY

2.1 MODELS

We will focus on the world model learning aspect of multi-view RL. Given a good world model, any existing model-based RL method could be employed to solve the original RL problem. We will thus assume that we are given a dataset of trajectories $(x_{0:T}, a_{0:T})$, collected by some policies.

The model components that we need to learn a world model are

- an emission model $p(x_t|z_t)$ to learn a mapping from the discrete latent space to the observation space
- a prior distribution over states $p(z_0)$
- an action-conditioned transition model $p(z_{t+1}|z_t, a_t)$
- a value-prefix predictor $p(vp|z_t, \gamma)$

We parameterize the prior distribution via a probability vector ρ_0, the action-conditioned transition model as a collection of transition matrices $\{T^a_{ij}\}_{a\in\mathcal{A}}$ and the emission model and value-prefix predictor via deep neural networks.

2.2 COMPUTING THE ROLLOUT PROBABILITIES

Given a sequence $(x_{0:T}, a_{0:T})$ we can compute the following quantities:

$$p(z_0|x_0) = \alpha_0(\rho_0 \cdot p(x_0|z_0))$$

And iteratively[3]

$$\begin{aligned} p(z_{t+1}|z_t, a_t) &= \mathbf{T}^{\mathbf{a_t}} p(z_t|z_{t-1}, a_{t-1}, x_t) \\ p(z_{t+1}|z_t, a_t, x_{t+1}) &= \alpha_{t+1} p(x_{t+1}|z_{t+1}) p(z_{t+1}|z_t, a_t), \end{aligned} \tag{1}$$

where α_t are normalization factors.

Based on this we can perform k step rollouts from each latent state z_t to compute estimates of the value prefix at each point in time.

If we go to the multi-view setting we can replace $p(x_{t+1}|z_{t+1})$ with $\prod_i p(x^i_{t+1}|z_{t+1})$, according to our assumption about the factorization of views conditional on the latent state.

2.2.1 UNOBSERVED VIEWS

The explicit update rule eq. (1) admits a particularly straight forward way of adopting to missing views. Namely, we can simply drop the factor $p(x^i_{t+1}|z_{t+1})$ of the view i that is not observed. This is in stark contrast to previous approaches which do not admit this solution.

2.3 VALUE PREFIX

As argued by Ye et al. (2021), it is sufficient for solving the RL problem to predict the value prefix, i.e. the sum of discounted rewards. This has the advantage of being less susceptible to state-aliasing and sparse rewards.

Given the current state z_t, Ye et al. (2021) propose to use the current estimate of the dynamics to perform a k step rollout, resulting in priors $\hat{z}_{t+i}$ $(i = 1, \ldots, k-1)$. They then feed this sequence into an LSTM which predicts the value prefixes for the z_{t+i}. The scalar value prefix targets are transformed as proposed by Schrittwieser et al. (2020) into categorical targets, resulting in a classification problem.

[3]Note that we omit older dependencies than the last step

$$\widehat{vp}(z_t; k) = f(z_t, \hat{z}_{t+1}, \ldots, \hat{z}_{t+k-1})$$

Here, $\widehat{vp}(z_t; k)$ is a sequence of predicted value prefixes, generated by an LSTM f which takes as input the true latent state z_t and the predicted next k latent states.

2.4 Loss components

2.4.1 Prior over states

The prior probabilities are trained towards the probability of the latent state after observing the first observation, i.e. its target is $p(z_0|x_0, \rho_0)$. Since minimizing KL divergence is equivalent to maximizing log-likelihood, we obtain as a loss function the cross entropy loss, where we apply the `stop_grad` operator on the posterior probabilities:

$$\mathcal{L}_{\text{prior}}(x_0) = -\sum_k \log \rho_{0k} \cdot \texttt{stop\_grad}\left[\alpha_0(\rho_0 \cdot p_\theta(x_0|z_0))\right]$$

The `stop_grad` is used to only pull the prior towards the posterior and not vice versa.

2.4.2 Emission Model

Given a trajectory $(x_{0:T}, a_{0:T})$, the emission model is trained via a reconstruction loss given the latent distribution predicted by the overall model:

$$\begin{aligned}
\mathcal{L}_{\text{Em}}(x_{0:T}) &= \sum_{t=0}^{T} \mathrm{D}_{\mathrm{KL}}\left(p(x_t) \| p_\theta(x_t|x_{0:t-1})\right) \\
&= \sum_{t=0}^{T} \mathop{\mathbb{E}}_{p(x_t)}\Big[-\log p_\theta(x_t|x_{0:t-1})\Big] + \text{const.} \\
&= \sum_{t=0}^{T} \mathop{\mathbb{E}}_{p(x_t)}\Big[-\log \sum_{z_t} p_\theta(x_t|z_t) p_\theta(z_t|x_{0:t-1})\Big] + \text{const.}
\end{aligned}$$

Since we do not have access to the true $p(x_t)$, we can approximate the expected value with a single Monte Carlo sample from the trajectory data.

$$\mathcal{L}_{\text{Em}}(x_{0:T}) \approx \sum_{t=0}^{T} -\log \sum_{z_t} p_\theta(x_t|z_t) p_\theta(z_t|x_{0:t-1}) + \text{const.}$$

2.4.3 Value-prefix predictor

Let the LSTM output on the sequence $(z_t, \hat{z}_{t+1}, \ldots, \hat{z}_{t+\ell-1})$ be $\widehat{vp}(z_t; \ell)$ and the target value prefix sequence be $vp(z_{t:t+\ell-1})$. Then the value-prefix loss on an observed trajectory is given by

$$\mathcal{L}_{\text{VP}}(x_{0:T}, a_{0:T}) = \sum_{t=0} \mathrm{D}_{\mathrm{KL}}\left(vp(z_{t:t+\ell-1}) \| \widehat{vp}(z_t; \ell)\right),$$

which can be easily computed in the case of discrete distributions.

Since we observe and predict sequences of distributions in latent space, rather than specific states z_t, we will have to adapt the algorithm slightly. Two options present themselves. We can either work on a parameterization of the distribution (e.g. the factorized distribution over latent variables), or we perform MCMC sampling. In our first attempts we will use the factorized distribution approach as it

does not discard any information. If this proves computationally intractable we will switch to the stochastic approach. We note that Hafner et al. (2020)[4] reported that they needed both a hidden state representing the distribution and also a stochastic sample to achieve good performance.

2.4.4 Dynamics Model

For the dynamics model, we want to train the transition model towards the posterior after observing the next observation. To gain a richer training signal we can also perform multiple step rollouts and train towards the k next posterior belief states.

If we only perform one-step rollouts, the dynamics loss would thus be given by

$$\mathcal{L}_{\text{Dyn}}(x_{0:T}, a_{0:T}) = \sum_{t=0}^{T} \mathrm{D}_{\mathrm{KL}}\left(p(z_t|x_{0:t}, a_{0:t}) \| p(z_t|x_{0:t-1}, a_{0:t})\right)$$

As proposed by Hafner et al. (2020), we perform KL-balancing, i.e. training prior and posterior with different coefficients. This can be accomplished via the `stop_grad` operator:

$$\begin{aligned}\mathcal{L}_{\text{Dyn}}(x_{0:T}, a_{0:T}) = \sum_{t=0}^{T} &\beta\, \mathrm{D}_{\mathrm{KL}}\left(\texttt{stop\_grad}[p(z_t|x_{0:t}, a_{0:t})] \| p(z_t|x_{0:t-1}, a_{0:t})\right) \\ &+ (1-\beta)\, \mathrm{D}_{\mathrm{KL}}\left(p(z_t|x_{0:t}, a_{0:t}) \| \texttt{stop\_grad}[p(z_t|x_{0:t-1}, a_{0:t})]\right)\end{aligned}$$

Hafner et al. (2020) chose $\beta = 0.8$ in their experiments. We note again that the difference between the prior and posterior is simply the observation x_t, meaning that in essence we train the transition matrix to anticipate the update from observing x_t.

2.4.5 Total Loss

Putting everything from above together, we can write the total world model loss for one complete trajectory as follows:

$$\begin{aligned}\mathcal{L}_{\text{tot}} = \sum_{t=0}^{T} &- \sum_{k} \log \rho_{0k} \cdot \texttt{stop\_grad}\left[\alpha_0(\rho_0 \cdot p_\theta(x_0|z_0))\right] \\ &- \sum_{t=0}^{T} \log \sum_{z_t} p_\theta(x_t|z_t) p_\theta(z_t|x_{0:t-1}) \\ &+ \mathrm{D}_{\mathrm{KL}}\left(vp(z_{t:t+\ell-1}) \| \widehat{vp}(z_t; \ell)\right) \\ &+ \beta\, \mathrm{D}_{\mathrm{KL}}\left(\texttt{stop\_grad}[p_\theta(z_t|x_{0:t}, a_{0:t})] \| p_\theta(z_t|x_{0:t-1}, a_{0:t})\right) \\ &+ (1-\beta)\, \mathrm{D}_{\mathrm{KL}}\left(p(z_t|x_{0:t}, a_{0:t}) \| \texttt{stop\_grad}[p_\theta(z_t|x_{0:t-1}, a_{0:t})]\right)\end{aligned} \tag{2}$$

[4] Not sure if this was the paper or one of their earlier ones

2.4.6 ALGORITHM

Algorithm 1 Training the world model

Require: Dataset $\mathcal{D}$ of trajectories $\tau = \{x_0, a_0, x_1, \ldots, a_{T-1}, x_T\}$.

Initialize world model parameters θ.
while not done **do**
  sample $\tau_i \sim \mathcal{D}$
  Using eq. (1) and τ_i, compute the belief over the latent states $z_{0:T}$
  Compute $\mathcal{L}_{\text{tot}}$ from eq. (2).
  Use backpropagation and SGD/Adam to update θ
end while

3 PLANNING

3.1 EXPERIMENTS

3.1.1 WORLD MODEL LEARNING

Simple Crossing Environment

To test our general approach and setup, we will conduct an initial toy experiment on the 'MiniGrid-SimpleCrossingS9N1-v0' environment[5]. The environment consists of a 7×7 grid world in which the agent has to navigate to the goal position in the bottom right. The agent starts out in the top left corner. The environment contains some grid cells that are blocked by walls. The position of the walls is randomly sampled upon resetting the environment. The agent receives a reward of 1 upon reaching the goal tile, and a constant reward of 0 otherwise.

First, we will only work with a single view, to determine whether our approach can work in the full-information setting.

As a baseline we are using a PyTorch DreamerV2 implementation[6].

The metrics that we will compare are:

- Average reconstruction loss of the observation model
- L1 loss of the value-prefix (ours) vs. L1 loss of the reward (DreamerV2)
- Reconstruction/Latent loss as a function of rollout steps, i.e. how good are the models at extrapolating

Once this preliminary experiment is concluded, we will add multiple views. We construct three views which each contain a random subset of 50% of the wall tiles. Each wall tile is shown in at least one view. Each view shows the goal and player position. Each view has an independent chance of 10% of just returning a null value instead of the actual view, simulating a faulty sensor. An example of all views is shown in fig. 1. Note in particular that all views share some relevant information but also potentially have relevant information that is unique to them.

Since DreamerV2 does not offer a principled multi-view approach, we simply concatenate the individual views, filling faulty views with a fixed null value. Again, we will compare the models along the aformentioned metrics.

Atari

The next step is to evaluate the models on the Atari Benchmark. Approach is going to be similar to toy experiment

[5] https://github.com/maximecb/gym-minigrid
[6] https://github.com/RajGhugare19/dreamerv20

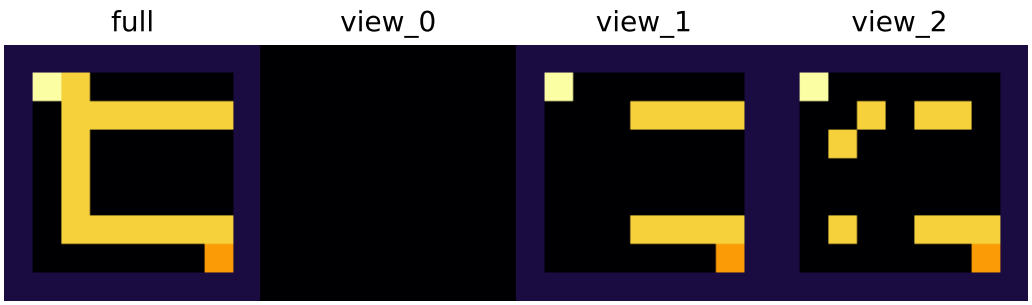

Figure 1: Simple crossing environment.

3.1.2 Behavior Learning

Once we have demonstrated good performance on the task of learning a world model from collected experience, we need to think about how to incorporate the task of learning the policy. On the one hand, Hafner et al. (2020) use the learned latent MDP to train via an actor-critic approach, either via reinforce gradients or via backpropagating through the dynamics.[7]

On the other hand, we could pursue the MCTS based approach of Schrittwieser et al. (2020), which has recently demonstrated tremendous data efficiency on the Atari benchmark if combined with self-consistency, value-prefix prediction and model-based re-analysis (Ye et al., 2021).

From an implementation point of view, the approach of Hafner et al. (2020) seems easier, so we might opt for that. It also allows us to train the world model separately from the actor and critic, whereas the MCTS based approach would use and end-to-end training approach.

3.2 Time Table

Item	End of April	May	June	July
Toy task::Ours	✓			
Toy task::Single	✓			
Toy task::Multi	✓			
Toy task::Write up		✓		
Atari::Setup		✓		
Atari::Ours		✓		
Atari::DreamerV2			✓	
Atari::Single			✓	
Atari::Multi			✓	
Atari::Write up			✓	
Final writeup				✓

Table 1: Time table. Checkmarks denote the end of which month the item should be finished. The model names (Ours, DreamerV2) correspond to making sure the method runs at all. Single and Multi correspond to full test suites using a single-view/multi-view with dropout respectively.

For making our method work on the more difficult environments like Atari, we will have to think about ways of scaling the HMM approach. We already have some thoughts on structuring and sparsifying the latent space to achieve this goal but it is not in a sufficiently detailed form yet to write it up properly.

References

Sudeep Fadadu, Shreyash Pandey, Darshan Hegde, Yi Shi, Fang-Chieh Chou, Nemanja Djuric, and Carlos Vallespi-Gonzalez. Multi-view fusion of sensor data for improved perception and prediction

[7] Don't quite understand the second part yet.

in autonomous driving, 2021.

Jiameng Fan and Wenchao Li. Dribo: Robust deep reinforcement learning via multi-view information bottleneck, 2021.

Marco Federici, Anjan Dutta, Patrick Forré, Nate Kushman, and Zeynep Akata. Learning robust representations via multi-view information bottleneck, 2020.

David Ha and Jürgen Schmidhuber. World models. *CoRR*, abs/1803.10122, 2018. URL http://arxiv.org/abs/1803.10122.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to Control: Learning Behaviors by Latent Imagination. pp. 1–20, 2019a. ISSN 2331-8422. URL http://arxiv.org/abs/1912.01603.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels, 2019b.

Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering Atari with Discrete World Models. pp. 1–26, 2020. URL http://arxiv.org/abs/2010.02193.

Minne Li, Lisheng Wu, Haitham Bou Ammar, and Jun Wang. Multi-view reinforcement learning, 2019.

Sherjil Ozair, Yazhe Li, Ali Razavi, Ioannis Antonoglou, Aäron van den Oord, and Oriol Vinyals. Vector quantized models for planning, 2021.

Tong Qin, Shaozu Cao, Jie Pan, and Shaojie Shen. A general optimization-based framework for global pose estimation with multiple sensors, 2019.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy Lillicrap, and David Silver. Mastering Atari, Go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020. ISSN 14764687. doi: 10.1038/s41586-020-03051-4. URL https://arxiv.org/pdf/1911.08265.pdf.

Qi Wang, Claire Boudreau, and Pang-ning Tan. Deep Multi-view Information Bottleneck. pp. 37–45, 2019. URL https://epubs.siam.org/doi/pdf/10.1137/1.9781611975673.5.

Weirui Ye, Shaohuai Liu, Thanard Kurutach, Pieter Abbeel, and Yang Gao. Mastering atari games with limited data, 2021.

Jing Zhao, Xijiong Xie, Xin Xu, and Shiliang Sun. Multi-view learning overview: Recent progress and new challenges. *Information Fusion*, 38:43–54, 2017. ISSN 1566-2535. doi: https://doi.org/10.1016/j.inffus.2017.02.007. URL https://www.sciencedirect.com/science/article/pii/S1566253516302032.