Backward Learning for Goal-Conditioned Policies

Marc Höftmann, Jan Robine, Stefan Harmeling
Department of Computer Science, Technical University of Dortmund, Germany

Abstract

Can we learn policies in reinforcement learning without rewards? Can we learn a policy just by trying to reach a goal state? We answer these questions positively by proposing a multi-step procedure that first learns a world model that goes backward in time, secondly generates goal-reaching backward trajectories, thirdly improves those sequences using shortest path finding algorithms, and finally trains a neural network policy by imitation learning. We evaluate our method on a deterministic maze environment where the observations are 64×64 pixel bird's eye images and can show that it consistently reaches several goals.

Keywords: World Models, Goal-conditioned, Reward-free

1 Motivation

Recently, generative auto-regressive models have shown great prospect with the developments around GPT-3 (Brown et al., 2020) where predictive coding (Huang & Rao, 2011) seems a promising avenue for further advancements in machine learning. Model-based Reinforcement learning (MBRL) applies the same principle already for decades (Schmidhuber, 1991) in order to learn the dynamics of an environment which is then exploited to maximize the return by learning a policy. These models can have different advantages to facilitate policy optimization, e.g., simulate future outcomes (Kaiser et al., 2019; Hafner et al., 2023; Robine et al., 2023), conduct online planning (Schrittwieser et al., 2020) or provide the model's state memory as an additional policy input (Ha & Schmidhuber, 2018).

But sometimes no reward function is given and instead some goal states have to be reached. An important variant of reinforcement learning (RL) is *goal-conditioned* RL, where a policy tries to arrive at a state configuration. Typical modern approaches (Andrychowicz et al., 2017; Hafner et al., 2022; Pateria et al., 2021; Ecoffet et al., 2021; Li et al., 2022; Gupta et al., 2019) start at a certain state and then try to reach a selected goal. We call this paradigm *hit-and-miss*, since the trajectories generated by these approaches within a reasonable amount of steps are not guaranteed to contain a desired goal.

Outline. In our work, we overcome this problem with the help of world models by ensuring that every produced trajectory *does* reach the goal state. We do this by starting at the goal and then going backward in time. To achieve this, we collect random forward trajectories on which we train a backwards world model that takes a state-action pair (s_t, a_{t-1}) as input and outputs the previous state, e.g., $f(s_t, a_{t-1}) = s_{t-1}$.

Once the backwards world model is sufficiently trained, we use it to learn policies. In goal-conditioned RL the assumption is that we are given some goal state s^*. In real world problems this goal might be problem specific or be determined by intrinsic or extrinsic reward, game score or by human preference. After the goal selection our backwards world model generates rollouts, reversed in time, that ultimately look like typical forward sequences $(s_0, a_0, \ldots, s_{t-1}, a_{t-1}, s_t, a_t, s_{t+1}, a_{t+1}, \ldots, s^*)$ but without rewards.

37th Conference on Neural Information Processing Systems (NeurIPS 2023) - Workshop on Goal-Conditioned Reinforcement Learning.

In principle these forward trajectories seem ideal to train a policy by imitation learning, since each sequence always reaches the goal state. However, the simulations should not be carelessly used for optimization due to potential loops or poor action selection (investigated by Wu et al., 2019; Wang et al., 2021b). Therefore, further tricks are needed to exclude some of the state-actions pairs, which we analyze and determine by graph search over simulated trajectories.

2 Prior Work

Offline RL. Chebotar et al. (2021) learn a goal-conditioned Q-function on offline data where they use sub-sequences and relabeling techniques to generate numerous of goal trajectories to learn from and failure actions to avoid. Kidambi et al. (2020) use offline MBRL to learn a pessimistic estimate of the underlying MDP and optimize a policy on it while minimizing the model bias.

Backward world models. Currently, world models that unroll simulated trajectories into the past (backward models) represent a small part in the model-based reinforcement learning literature. It starts with models that improve sample efficiency (Goyal et al., 2018) or use improved strategies for returning to high value states (Edwards et al., 2018). Lai et al. (2020) proposes bi-directional world models which generate short forward and backward trajectories to be more robust against prediction error of forward models that unroll for too many timesteps. Wang et al. (2021a) learn a reverse policy and reverse dynamics model for offline reinforcement learning. They generate backward simulations which lead to novel sub-goal reaching trajectories with respect to the existing offline dataset. Chelu et al. (2020) address the RL credit assignment problem and examine the benefits of planning with a backward model in relationship to forward models, where they observe complementary gains using backward models. Pan & Lin (2022) combine imitation learning with backward world models where backward simulations are treated as sub-optimal expert demonstrations to improve the expected return of a forward policy. In this work, we utilize our discrete backward model in a way which allows it to create demonstrations that can be used without considering the expected return.

3 Method

As commonly done in model-based RL, we project the raw state observations into a latent space. In order to also utilize return strategies from Go-Explore methods (Ecoffet et al., 2021; Höftmann et al., 2023) we learn an encoder Φ_θ that generates a *discrete* latent representation (following Hafner et al., 2020) jointly with a *latent* world model Ψ_θ. In short, we denote them as

$$\Phi_\theta(s_t) = Z_t \qquad\qquad \Psi_\theta(z_t, a_{t-1}) = \hat{Z}_{t-1}. \tag{1}$$

where $Z_t, \hat{Z}_{t-1}$ are distributions over the latent space and discrete samples z_t can be obtained via sampling, i.e., $z_t \sim Z_t$. In addition, we define the latent space decoder as $s_t \sim p_\theta(s_t|z_t)$.

Representation Model. The representation model is implemented by a Convolutional Neural Network (CNN, LeCun et al., 1989) with an Encoder-Decoder architecture (Ballard, 1987) that uses the categorical latent representation from Hafner et al. (2020) with straight-through gradients. In our case, the encoder's latent representation $Z \in \mathbb{R}^{g \times c}$ is stochastic and uses $g = 16$ categoricals with $c = 16$ classes each, where we obtain a discrete representation $z \sim Z$ by sampling from g softmax distributions. The representation loss can be written as

$$\mathcal{L}_\theta^{\text{Repr}} = \mathbb{E}_t \left[\underbrace{-\log p_\theta(s_t|z_t)}_{\text{reconstruction loss}} + \underbrace{\alpha \cdot \max\left(\frac{1}{gc} \sum_{i=1}^{g} \sum_{j=1}^{c} Z_{t_{i,j}} \log(Z_{t_{i,j}}),\ 0.05 \right)}_{\text{entropy loss}} \right] \text{ with } z_t \sim Z_t, \tag{2}$$

and consists of two objectives. The first one is the classic reconstruction loss to minimize the decoder's image reconstruction error using a binary cross entropy loss. Secondly, we add an entropy loss term with weighting α that increases over time. Its schedule is based on the current training

epoch e_c and the terminal training epoch e_T as follows

$$\alpha = \begin{cases} 0 & \text{if } e_c < 0.9 \cdot e_T, \\ 5 \times 10^{-6} & \text{otherwise,} \end{cases} \tag{3}$$

such that the entropy of our latent representation decreases at the end of training to a minimum. This is necessary, since each observation should be precisely projected to only one latent sample in order to use return strategies from Go-Explore (Ecoffet et al., 2021) but also to build directed graphs without multiple nodes for a single observation.

Dynamics Model. The backwards world model uses a Multi-Layer Perceptron (MLP) (Rosenblatt, 1958) with SiLU nonlinearity (Elfwing et al., 2018) and layer normalization (Ba et al., 2016). It takes an observation and action encoding (z_t, a_{t-1}) as input and outputs the estimated distribution $\hat{Z}_{t-1}$ over previous latent states. Consequently, the world model's loss minimizes the discrete Kullback–Leibler ($\mathrm{D_{KL}}$) divergence

$$\begin{aligned} \mathcal{L}_\theta^{\text{WM}} = \; & \mathbb{E}_t \Big[\mathrm{D_{KL}}\big(\hat{Z}_{t-1} \| Z_{t-1}\big) \Big], \\ & \text{with } \; \Phi_\theta(s_{t-1}) = Z_{t-1}, \\ & \text{and } \; \Psi_\theta(z_t, a_{t-1}) = \hat{Z}_{t-1} \end{aligned} \tag{4}$$

between the encoder's latent distribution Z_{t-1} and the world model's predicted distribution $\hat{Z}_{t-1}$. Both models are jointly trained to allow gradient flow back into the encoder parameters and therefore allowing the world model to influence the encoding structure in the latent space. The complete loss function is written as

$$\mathcal{L}_\theta = \mathcal{L}_\theta^{\text{Repr}} + w_{\text{wm}} \cdot \mathcal{L}_\theta^{\text{WM}}, \tag{5}$$

which combines both terms with an additional weight $w_{\text{wm}} = 0.0025$ for the world model loss.

Policy optimization. With a *discrete* world model at hand, we create an improved dataset $\mathcal{D}$ for imitation learning instead of just relying on raw rollout trajectories from a (non-discrete) world model. The following procedure is implemented:

1. Generate backward simulations via random action selection while starting at the goal z^*. In addition, start further rollouts at already simulated states based on their inverse visit frequency (following the cell selection rule from Ecoffet et al., 2021).
2. Create a directed graph (DG) with the simulation data where each node is a state encoding z_t. A *directed* edge is added to the graph when a backward state transition (z_{t+1}, z_t) is found in the simulations. In that way, the DG contains and unifies the information from all separate simulations.
3. Apply a shortest path finding algorithm on the DG to find the shortest distances to the goal node. For our method we use Dijkstra's algorithm (Dijkstra, 1959) to get a shortest path estimate (SPE) for every state in the graph, e.g., $\text{SPE}(z^*) = 0$.

By using the SPE, we evaluate whether a state-action pair (z_t, a_t) should be used for the imitation learning dataset $\mathcal{D}$: if a forward transition (z_t, a_t, z_{t+1}) brings us closer to the goal, we include it, i.e.

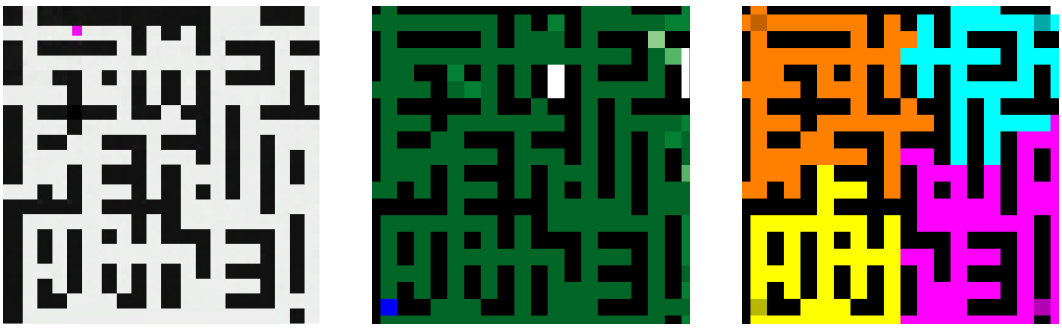
$$(\text{SPE}(z_t) > \text{SPE}(z_{t+1})) \Rightarrow (z_t, a_t) \in \mathcal{D}. \tag{6}$$

Note, that this dataset $\mathcal{D}$ has no loops and contains only paths that directly lead to the goal (according to the world model). In fact, the rule selects only shortest paths since it uses Dijkstra's algorithm for evaluation.

Finally we use the standard imitation learning technique of maximizing the log-probability for the selected state-action pairs (z_t, a_t), i.e., we minimize

$$\mathcal{L}_\theta^{\text{policy}} = \mathbb{E}_t \left[-\log(\pi_\theta(a_t|z_t)) + c_1 S[\pi_\theta](z_t) \right] \;\; \text{with } \; (z_t, a_t) \in \mathcal{D}, \tag{7}$$

where we added a small action space entropy bonus S (Schulman et al., 2017) with weight c_1. The entropy bonus is helpful if $\mathcal{D}$ contains bad samples. Furthermore, it makes additional policy improvement easier, e.g., by applying further classical reinforcement learning using REINFORCE (Sutton & Barto, 2018) on the policy π_θ.



(a) Maze (20×20) observation (b) Single goal policy (c) Four goal policy

Figure 1: Visualization of the maze environment. Panel (a) shows a raw observation where no goals are marked. Panel (b) shows the results for a learned policy with one goal state (marked as blue). The brightness of a square in green indicates how often we reached the goal. Panel (c) shows results for four goal states (marked as dark-{orange,cyan,yellow,violet}). Colors indicate to which goal the policy has returned to, e.g., a position is colored orange when the agent went to the dark-orange goal.

(a) One goal optimization

Maze size	Goal	Return positions
20 × 20	Orange	207.1 / 234 (88%)
	Cyan	218.5 / 234 (93%)
	Yellow	210.2 / 234 (90%)
	Violet	207.7 / 234 (89%)

(b) Four goal optimization

Maze size	Return positions	Went to closest goal
20 × 20	221.9 / 234 (95%)	219.9 / 221.9 (99%)

Figure 2: Quantitative evaluation on the maze shown in Figure 1 with 20 runs for each result. One and four goal optimization use the goals from Figure 1c.

4 Experiments

We study our method on a 20x20 maze environment where an observation is one 64×64 pixel image with RGB color channels and has a bird's-eye view over the whole maze (see Fig. 1a). Note, that a deterministic maze is *non-deterministic* when it is unrolled backwards in time. Our latent world model incorporates this property by predicting a latent state distribution (as in Hafner et al., 2023). For this work, the encoder, the decoder, the world model and the policy are neural networks.

After the world model training, every learned policy is conditioned on up to four goals. For our test, we choose the goal locations at the maze corners shown in Fig. 1b/c. To deal with multiple goals for one policy requires only tiny adjustments in the DG. Finally, all policies are tested by placing the agent at every maze position five times and check whether they can return to their goal. To exclude attempts that reach the goal in time by chance, we consider only trials to be successful, if a goal was reached within 1.5 times the shortest goal distance. The locations marked in lighter green or even white indicate that the agent did not always reach the goal (see Fig. 1b/c).

Fig. 2 shows the quantitative evaluation on the maze. Policies are conditioned either on one goal (bottom left corner in Fig. 1b) or on four goals simultaneously (all corners in Fig. 1c). The results show that the policies can reach their goal(s) from almost every maze position which are up to 37 steps away. Our experiments show we can learn complex goal-oriented policies even without rewards.

5 Conclusion

We introduce a novel method for goal-conditioning a policy on one or more states by using a discrete world model jointly with graph search. Even though our method is not using rewards for optimization, it is still able to reach the desired state configurations by using a refined imitation learning dataset that has been generated by our backward world model.

References

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Dana H Ballard. Modular learning in neural networks. In *Proceedings of the sixth National Conference on artificial intelligence-volume 1*, pp. 279–284, 1987.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Yevgen Chebotar, Karol Hausman, Yao Lu, Ted Xiao, Dmitry Kalashnikov, Jake Varley, Alex Irpan, Benjamin Eysenbach, Ryan Julian, Chelsea Finn, et al. Actionable models: Unsupervised offline reinforcement learning of robotic skills. *arXiv preprint arXiv:2104.07749*, 2021.

Veronica Chelu, Doina Precup, and Hado P van Hasselt. Forethought and hindsight in credit assignment. *Advances in Neural Information Processing Systems*, 33:2270–2281, 2020.

Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1 (1):269–271, 1959.

Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune. First return, then explore. *Nature*, 590(7847):580–586, 2021.

Ashley D Edwards, Laura Downs, and James C Davidson. Forward-backward reinforcement learning. *arXiv preprint arXiv:1803.10227*, 2018.

Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural networks*, 107:3–11, 2018.

Anirudh Goyal, Philemon Brakel, William Fedus, Soumye Singhal, Timothy Lillicrap, Sergey Levine, Hugo Larochelle, and Yoshua Bengio. Recall traces: Backtracking models for efficient reinforcement learning. *arXiv preprint arXiv:1804.00379*, 2018.

Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. *arXiv preprint arXiv:1910.11956*, 2019.

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020.

Danijar Hafner, Kuang-Huei Lee, Ian Fischer, and Pieter Abbeel. Deep hierarchical planning from pixels. *Advances in Neural Information Processing Systems*, 35:26091–26104, 2022.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

Marc Höftmann, Jan Robine, and Stefan Harmeling. Time-myopic go-explore: Learning a state representation for the go-explore paradigm. *arXiv preprint arXiv:2301.05635*, 2023.

Yanping Huang and Rajesh PN Rao. Predictive coding. *Wiley Interdisciplinary Reviews: Cognitive Science*, 2(5):580–593, 2011.

Lukasz Kaiser, Mohammad Babaeizadeh, Piotr Milos, Blazej Osinski, Roy H Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, et al. Model-based reinforcement learning for atari. *arXiv preprint arXiv:1903.00374*, 2019.

Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel: Model-based offline reinforcement learning. *Advances in neural information processing systems*, 33: 21810–21823, 2020.

Hang Lai, Jian Shen, Weinan Zhang, and Yong Yu. Bidirectional model-based policy optimization. In *International Conference on Machine Learning*, pp. 5618–5627. PMLR, 2020.

Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

Jinning Li, Chen Tang, Masayoshi Tomizuka, and Wei Zhan. Hierarchical planning through goal-conditioned offline reinforcement learning. *IEEE Robotics and Automation Letters*, 7(4):10216–10223, 2022.

Yuxin Pan and Fangzhen Lin. Backward imitation and forward reinforcement learning via bi-directional model rollouts. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 9040–9047. IEEE, 2022.

Shubham Pateria, Budhitama Subagdja, Ah-hwee Tan, and Chai Quek. Hierarchical reinforcement learning: A comprehensive survey. *ACM Computing Surveys (CSUR)*, 54(5):1–35, 2021.

Jan Robine, Marc Höftmann, Tobias Uelwer, and Stefan Harmeling. Transformer-based world models are happy with 100k interactions. *arXiv preprint arXiv:2303.07109*, 2023.

Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

Jürgen Schmidhuber. A possibility for implementing curiosity and boredom in model-building neural controllers. In *Proc. of the international conference on simulation of adaptive behavior: From animals to animats*, pp. 222–227, 1991.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Jianhao Wang, Wenzhe Li, Haozhe Jiang, Guangxiang Zhu, Siyuan Li, and Chongjie Zhang. Offline reinforcement learning with reverse model-based imagination. *Advances in Neural Information Processing Systems*, 34:29420–29432, 2021a.

Yunke Wang, Chang Xu, Bo Du, and Honglak Lee. Learning to weight imperfect demonstrations. In *International Conference on Machine Learning*, pp. 10961–10970. PMLR, 2021b.

Yueh-Hua Wu, Nontawat Charoenphakdee, Han Bao, Voot Tangkaratt, and Masashi Sugiyama. Imitation learning from imperfect demonstration. In *International Conference on Machine Learning*, pp. 6818–6827. PMLR, 2019.