Non-Markovian Reward Modelling from Trajectory Labels via Interpretable Multiple Instance Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

We generalise the problem of reward modelling (RM) for reinforcement learning (RL) to handle non-Markovian rewards. Existing work assumes that human evaluators observe each step in a trajectory independently when providing feedback on agent behaviour. In this work, we remove this assumption, extending RM to include hidden state information that captures temporal dependencies in human assessment of trajectories. We then show how RM can be approached as a multiple instance learning (MIL) problem, and develop new MIL models that are able to capture the time dependencies in labelled trajectories. We demonstrate on a range of RL tasks that our novel MIL models can reconstruct reward functions to a high level of accuracy, and that they provide interpretable learnt hidden information that can be used to train high-performing agent policies.

1 Introduction

There is growing consensus around the view that aligned and beneficial AI requires a reframing of objectives as being contingent, uncertain, and learnable via interaction with humans [31]. In reinforcement learning (RL), this proposal has found one formalisation in reward modelling (RM): the inference of agent objectives from human preference information such as demonstrations, pairwise choices, approval labels, and corrections [25]. The objective of RM is to reconstruct an underlying *reward function*, and it is typically assumed that a human evaluates the *return* (quality) of a temporally-extended trajectory of agent behaviour by summing equal and independent *reward* assessments of instantaneous states and actions. However, in reality the human's experience of a trajectory is likely to be temporally extended (e.g., via a video clip [12] or real-time observation), which opens the door to dependencies between earlier events and the assessment of later ones. The independence assumption may be both psychologically unrealistic given human memory limitations [23], and technically naïve given the difficulty of building complete instantaneous state representations [22]. We thus seek to generalise RM to allow for temporal dependencies in human evaluation, by postulating *hidden state* information that accumulates over a trajectory. Reconstruction of the human's preferences now requires the modelling of hidden state dynamics alongside the reward function itself.

In tackling this generalised problem, we identify a structural isomorphism between RM (specifically from trajectory return labels) and the established field of multiple instance learning (MIL) [9]. Trajectories are recast as *bags* and constituent state-action pairs as *instances*, which collectively contribute to labels provided at the bag level by interacting in potentially complex ways. This mapping inspires a range of novel MIL model architectures that use long short-term memory (LSTM) modules [16] to recover the hidden state dynamics, and learn instance-level reward predictions from return-labelled trajectories of arbitrary length. In experiments with synthetic oracle labels, we show that our novel MIL RM models can accurately reconstruct ground-truth hidden states and reward functions for non-Markovian tasks, and can be straightforwardly integrated into RL agent training to

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

achieve performance matching, or even exceeding, that of agents with direct access to true hidden states and rewards. Our contributions are as follows:

1. We generalise RM to handle *non-Markovian* rewards that depend on hidden features of the environment or the psychology of the human evaluator in addition to visible states/actions.
2. We identify a structural connection between RM and MIL, creating the opportunity to transfer concepts and methods between the two fields.
3. We propose two novel LSTM-based MIL models for this generalised RM problem, and develop interpretability techniques for understanding and verifying the learnt reward functions.
4. We compare our proposed models to existing MIL baselines on four non-Markovian tasks, evaluating return prediction, reward prediction, robustness to label noise, and interpretability.
5. We demonstrate that the hidden state and reward predictions of our MIL RM models can be used by RL agents to solve non-Markovian tasks.

The remainder of this work is as follows. Section 2 discusses related work in RM and MIL, Section 3 gives a formal problem definition and describes our MIL-inspired methodology, and Section 4 presents our experiments and results. We discuss key findings in Section 5, and Section 6 concludes.

2 Background and Related Work

Reward Modelling RM [25] aims to infer a reward function from revealed human preference information such as demonstrations [28], pairwise choices [12], corrections [4], good/bad/neutral labels [29], or combinations thereof [21]. Most prior work assumes a human evaluates a trajectory by summing independent rewards for each state-action pair, but in practice their experience is likely to be temporally extended (e.g., via a video clip), creating the opportunity for dependencies to emerge between earlier events and the assessment of later ones. As noted by Chan et al. [10] and Bewley and Lecue [7], such dependencies may arise from cognitive biases such as anchoring, prospect bias, and the peak-end rule [23], but they could equally reflect rational drivers of human preferences not captured by the state representation. Some efforts have been made to model temporal dependencies, such as a discrete psychological mode which evolves over consecutive queries about hypothetical trajectories [6], or a monotonic bias towards more recently-viewed timesteps due to human memory limitations [24]. Elsewhere, Shah et al. [32] use human demonstrations and binary approval labels to learn temporally-extended task specifications in logical form. In comparison to these restricted examples, our work provides a more general approach to capturing temporal dependencies in RM.

Non-Markovian Rewards In the canonical RL problem setup of a Markov decision process (MDP), rewards depend only on the most recent state-action pair. In a non-Markovian reward decision process (NMRDP) [2], rewards depend on the full preceding trajectory [2]. NMRDPs can be *expanded* into MDPs (and thus solved by RL) by augmenting the state with a hidden state that captures all reward-relevant historical information, but this is typically not known *a priori*. Data-driven approaches to learning NMRDP expansions [18] often make use of domain-specific propositions and temporal logic operators [3, 35, 36]. Outside of the RM context, recurrent architectures such as LSTMs have been used in NMRDPs to reduce reliance on pre-specified propositions [20]. They also have a long history of use in partially observable MDPs, where dynamics are also non-Markovian [5, 41, 14].

Multiple Instance Learning In MIL [9], datasets are structured as collections of bags $X_i \in \mathbf{X}$, each of which is comprised of instances $\{x_1^i, \ldots, x_k^i\}$ and has an associated bag-level label Y_i and instance-level labels $\{y_1^i, \ldots, y_k^i\}$. The aim is to construct a model that learns solely from bag labels; instance labels are not available during training, but may be used later to evaluate instance-level predictions. The simplest MIL approaches assume that instances are independent and that the bag is unordered, but models exist for capturing various types of instance dependencies [40, 19, 37]. LSTMs have emerged as a natural architecture for modelling temporal dependencies among ordered bags, where they can be utilised to aggregate instance information into an overall bag representation. They have previously been applied to standard MIL benchmarks [39], as well as specific problems such as Chinese painting image classification [26]. As we discuss in Section 3.2, these existing models are somewhat unsuitable for use in RM, leading us to propose our own novel model architectures.

3 Methodology

In this section, we present the core methodology of our work. We formally define the new paradigm of non-Markovian RM (Section 3.1), before drawing on the MIL literature to propose models that can be used to solve this generalised problem (Section 3.2). We then go on to discuss how we can use our learnt RM models for training RL agents on non-Markovian tasks (Section 3.3).

3.1 Formal Definition of Non-Markovian RM

Consider an agent interacting with an environment with Markovian dynamics. At discrete time t, the current environment state $s_t \in \mathcal{S}$ and agent action $a_t \in \mathcal{A}$ condition the next environment state s_{t+1} according to the dynamics function $D : \mathcal{S} \times \mathcal{A} \to \Delta\mathcal{S}$. A trajectory $\xi \in \Xi$ is a sequence of state-action pairs, $\xi = ((s_0, a_0), ..., (s_{T-1}, a_{T-1}))$, and a human's preferences about agent behaviour respect a real-valued return function $G : \Xi \to \mathbb{R}$. In traditional (Markovian) RM, return is assumed to decompose into a sum of independent rewards over state-action pairs, $G(\xi) = \sum_{t=0}^{T-1} R(s_t, a_t)$, and the aim is to reconstruct $R' \approx R$ from possibly-noisy sources of preference information. In our generalised non-Markovian model, we consider the human to observe a trajectory sequentially and allow for the possibility of hidden state information that accumulates over time and parameterises R:

$$G(\xi) = \sum_{t=0}^{T-1} R(s_t, a_t, h_{t+1}) \quad \text{where} \quad h_{t+1} = \delta(h_t, s_t, a_t), \tag{1}$$

δ is a hidden state dynamics function, and h_0 is a fixed value for the initial hidden state. Reconstruction of the human's preferences now requires the estimation of $\delta' \approx \delta$ and $h'_0 \approx h_0$ alongside $R' \approx R$. We can interpret the hidden information as an external feature of the environment that is visible to the human but excluded from the state, or as a psychological state of the person themselves, through which their response to a new observation is influenced by what they have seen already. Appendix B elaborates on this discussion, presenting motivating use cases and limitations of non-Markovian RM. We visualise the difference between Markovian and non-Markovian RM in Figure 1.

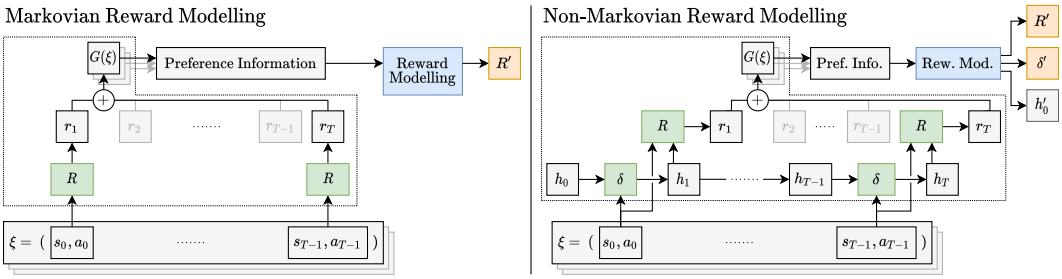

Figure 1: In Markovian RM, the human is assumed to sum over independent and equal assessments of the state-action pairs in a trajectory. In non-Markovian RM, per-timestep rewards additionally depend on hidden state information h that accumulates over time.

In this work, we focus on the simplest and most explicit form of preference information: direct labelling of returns $G(\xi_i)$ for a dataset of N trajectories $\{\xi_i\}_{i=1}^N$. We aim to solve the reconstruction problem by minimising the squared error in predicted returns:

$$\underset{R', \delta', h'_0}{\text{argmin}} \sum_{i=1}^{N} \left(G(\xi_i) - \sum_{t=0}^{T_i-1} R'(s_{i,t}, a_{i,t}, h'_{i,t+1}) \right)^2 \text{ where } \begin{array}{l} h'_{i,0} = h'_0 \\ h'_{i,t+1} = \delta'(h'_{i,t}, s_{i,t}, a_{i,t}) \end{array} \forall i \in \{1..N\}. \tag{2}$$

We observe that Equation 2 perfectly matches the definition of a MIL problem. Each trajectory ξ_i can be considered as an ordered bag of instances $((s_{i,0}, a_{i,0}), ..., (s_{i,T_i-1}, a_{i,T_i-1}))$ with unobserved instance labels $R(s_{i,t}, a_{i,t}, h_{i,t+1})$, an observed bag label $G(\xi_i) = \sum_{t=0}^{T-1} R(s_{i,t}, a_{i,t}, h_{i,t+1})$, and temporal instance interactions via the changing hidden state $h_{i,t}$. This correspondence motivates us to review the space of existing MIL models (specifically those that model temporal dependencies among instances) to provide a starting point for developing our non-Markovian RM approach.

3.2 MIL RM Architectures

The MIL literature contains a variety of architectures for handling temporal instance dependencies, including graph neural networks (GNNs) [37] and transformers [34]. While effective for many problems, such architectures are an unnatural fit to non-Markovian RM as they contain no direct analogue of a hidden state h' carried forward in time, instead handling dependencies via some variant of message-passing between instances. LSTM-based MIL architectures [39, 26] provide a more promising starting point since they explicitly represent both h' (implemented as a continuous-valued vector) and its temporal dynamics function δ' (a particular arrangement of gating functions).

Starting from an existing LSTM-based MIL architecture, we propose two successive extensions as well as a naïve baseline that cannot handle temporal dependencies. All four architectures include a feature extractor (FE) for mapping state-action pairs into feature vectors and a head network (HN) that outputs predictions. These architectures are depicted in Figure 2. Note we use the same nomenclature as [9] and [40]: *embedding space* approaches produce an overall bag representation that is used for prediction, while *instance space* approaches produce predictions for each instance in the bag and then aggregate those predictions to a final bag prediction.

Base Case: Embedding Space LSTM This architecture, proposed by Wang et al. [39], processes all instances in a bag sequentially and uses the final LSTM hidden state as a bag embedding. This is fed into the HN, which predicts the bag label g' (return in the RM context). Although this model can account for temporal dependencies, it does not inherently produce instance predictions (rewards), which require some post hoc analysis to recover. While methods exist for computing instance importance values as a form of interpretability [13], these are not guaranteed to sum to the bag label as stipulated by the reward-return formulation. We propose a new method: at time t, the predicted reward r'_t is calculated by feeding the LSTM hidden state at times $t-1$ and t into the HN to obtain two *partial* bag labels/returns g'_{t-1} and g'_t, and computing the difference of the two, i.e., $r'_t = g'_t - g'_{t-1}$. Note that we define $g'_0 = 0$.

Extension 1: Instance Space LSTM The preceding post hoc computation is rather inelegant and often yields poor reward predictions (see Appendix C and Section 4), likely because rewards are never computed or back-propagated through during learning. This leads us to propose an improved architecture, which is structurally similar but differs in how network outputs are mapped onto RM concepts, and places reward predictions on the back-propagation path. Given the LSTM hidden state at time t, the output of the HN is taken to be the instantaneous reward r'_t rather than the partial return. Rewards are computed sequentially for all timesteps in a trajectory and summed to give the return prediction g'. We thereby obtain a model that both handles temporal dependencies and produces explicitly-learnt reward predictions.

Extension 2: Concatenated Skip Connection (CSC) Instance Space LSTM In both of the preceding architectures, the LSTM hidden state h'_t is the sole input to the HN. This requires h'_t to represent all reward-relevant information from both the true hidden state h_t and the latest state-action pair s_{t-1}, a_{t-1} to achieve good performance. To lighten the load on the LSTM, we further extend the Instance Space LSTM model with a skip connection [15, 17] which concatenates the FE output onto the hidden state before feeding it to the HN. In principle, this should allow the hidden state to solely focus on representing temporal dependencies. As well as improving RM performance compared to an equivalent model without skip connections, we find in Section 5.1 that this modification tends to yield more interpretable and disentangled hidden state representations.

Markovian Baseline: Instance Space Neural Network (NN) To quantify the cost of ignoring temporal dependencies, we also run experiments with a baseline architecture that feeds only the FE output for each state-action pair into the HN, yielding fully-independent reward predictions which are summed to give the return prediction. This independent predict-and-sum architecture has precedence in both MIL, where it is referred to as mi-Net by Wang et al. [40], and in RM, where it embodies the de facto standard Markovian reward assumption [12].

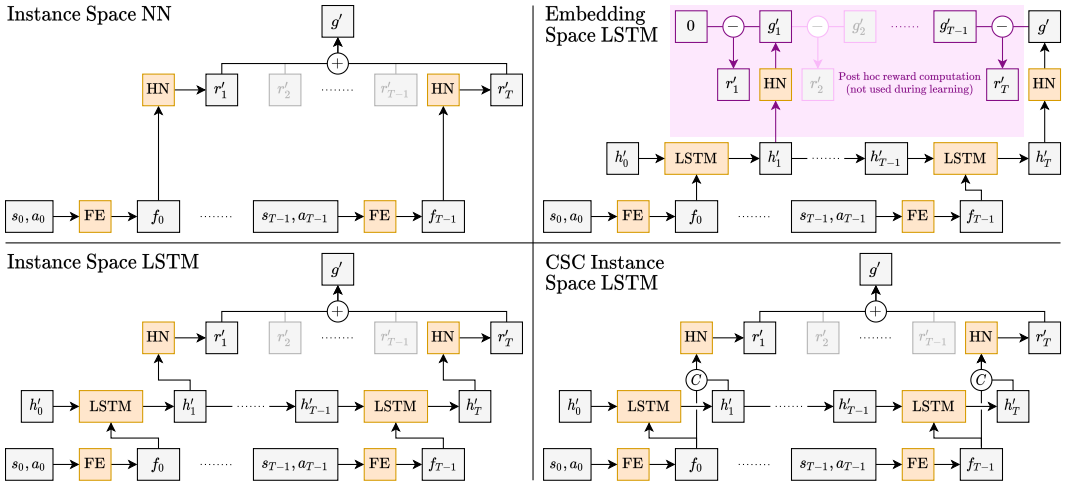

Figure 2: MIL architectures used in this work. FE = feature extractor; HN = head network; $(+)$ = scalar summation; $(-)$ = scalar subtraction; (C) = vector concatenation.

3.3 Training Agents with Non-Markovian RM Models

In this work, as in RM more widely, we are not solely interested in learning reward functions to represent human preferences, but also in the downstream application of rewards to train agents' action-selection policies. After optimising our LSTM-based models on offline trajectory datasets, we deploy them at the interface between conventional RL agents and their environments. Going beyond prior work, where a learnt model is used to either generate a reward signal for an agent to maximise [12] or augment its observed state representation with hidden state information [18], our models serve a dual role, providing *both* rewards and state augmentations. Figure 3 describes this setup in detail.

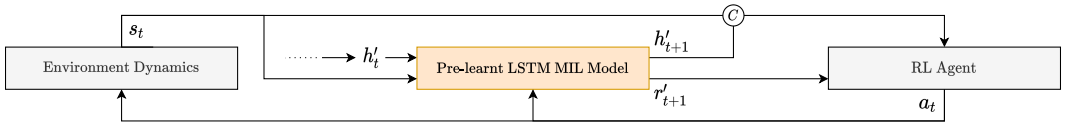

Figure 3: During RL agent training, our LSTM MIL models sit at the centre of the agent-environment loop by which states s_t and actions a_t are exchanged. We focus on episodic tasks, where the environment state periodically resets. The LSTM hidden state is simultaneously reset to h'_0 at the start of an episode, then is iteratively updated over time t given the state-action pairs s_t, a_t. At time t the environment state s_t is augmented with the post-update hidden state h'_{t+1} by concatenation, and this augmented state is observed by the agent. s_t, a_t and h'_{t+1} are used to compute a reward r'_{t+1} following the relevant steps from Figure 2, and the reward is also sent to the agent. In the language of NMRDPs, the hidden state augmentation *expands* the agent's learning problem into an MDP by providing the additional information required to make the rewards Markovian. Note that unlike during learning of the MIL models, return predictions are never required. Also note the importance of a recurrent model architecture. As they lack a sequentially-updated hidden state, transformer- or GNN-based MIL models would be unsuitable for such online deployment.

4 Experiments and Results

After initially validating our models on several toy datasets (see Appendix C), we focus the bulk of our evaluation on four RL tasks. As running experiments with people is costly, we use the standard RM approach of generating synthetic preference data (here trajectory return labels) using ground-truth *oracle* reward functions [12]. Unlike prior work, these oracle reward functions depend on historical information that cannot be recovered from the instantaneous environmental state, thereby emulating the disparity between the information that a human evaluator possesses while viewing a trajectory sequentially, and that contained in the state alone. In this section, we introduce our RL tasks (Section 4.1), evaluate the quality of reward reconstruction (Section 4.2), investigate the use of MIL RM models for agent training (Section 4.3), and evaluate their robustness to label noise (Section 4.4).

4.1 RL Task Descriptions

We propose four tasks with different types of hidden information. All use a 2D navigation environment with two spawn zones (where the agent initialises), and an episode time limit of $T = 100$; see Figure 4. In each case, the environment state contains the x, y position of the agent only. The tasks involve moving into a *treasure* zone, contingent on some hidden information that cannot be derived from the current x, y position, but is instead a function of the full preceding trajectory. In the first two cases the hidden information varies with time only, but in the other two it depends on the agent's past positions.

Timer For times $t \leq 50$ the treasure gives a reward of -1 for each timestep that the agent spends inside it, before switching to $+1$ thereafter. Since time is not included in the environment state, recovering the reward function by only observing the agent's current position is impossible.

Moving The Timer task only captures a binary change, therefore we generalise it to be continuous. In this case, the treasure zone oscillates left and right at a constant speed. Again, this is not captured in the environment state, but can be recovered if the length of the preceding trajectory is known.

Key Before reaching the treasure zone, the agent must first enter a second zone to collect a key; otherwise it receives 0 reward. As the key's status is not captured in the environment state, a temporal dependency exists between the agent's past positions and the reward it obtains from the treasure.

Charger We generalise the Key task by replacing the key zone with a charging zone that builds up the amount of reward the agent will receive when it reaches the treasure. Now, the reward is dependent on not only if the agent visits a zone (binary), but how long it is there (continuous).

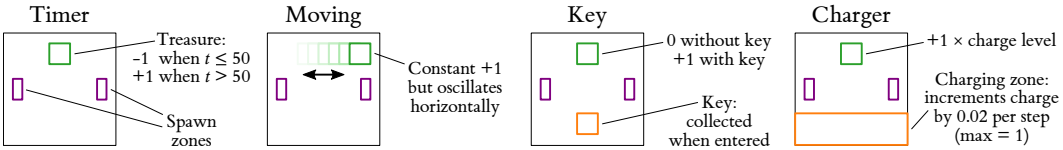

Figure 4: Environment layouts for the four non-Markovian RL tasks.

Further details on the tasks and MIL model hyperparameters are given in Appendix D. One important design decision for the LSTM-based models is the size of the hidden state, as it affects both performance and interpretability. For all the above tasks, we know *a priori* that it is possible to capture the temporal dependencies in at most two dimensions, therefore we constrain our models to use 2D hidden states. This allows us to visualise and interpret the hidden representations in Section 5.1.

4.2 Reward Modelling Results

Below we discuss the performance of the reward reconstruction for the different MIL RM models on our four RL tasks. For each task, we generate 5000 trajectories of length 100 by deploying a random agent and post-screening to obtain a wide distribution of outcomes and return values (see Appendix D for details on the screening process). Summary results from MIL models trained on these trajectories are given in Table 1. We observe the CSC Instance Space LSTM is on average the best performing model for predicting both trajectory returns and timestep rewards. While the Embedding Space LSTM model performs best at predicting return on the Key task, it struggles on the reward metric (as it relies on the use of a proxy post hoc method). Also note that the Instance Space NN that serves as our Markovian RM baseline performs very poorly on return prediction, indicating that these tasks indeed cannot be learnt without modelling temporal dependencies.

4.3 RL Training Results

Following the method from Section 3.3, we then experiment with training Deep Q-Network (DQN) [27] RL agents to optimise the rewards learnt by the LSTM-based models. We evaluate agent performance in a post hoc manner by passing its trajectories to the relevant oracle. Such an evaluation provides an end-to-end measure of both reward reconstruction and policy learning, and is standard in RM [12]. We baseline against agents trained with access to: a) the oracle reward function and the oracle hidden states, and b) just the oracle reward function without hidden states (i.e, using only the environment states that are missing information). From the results in Figure 5, we observe that the

Table 1: RL dataset return (top) and reward (bottom) results. Each measurement is the mean MSE averaged over ten repeats, with the standard errors of the mean also given.

Model	Timer	Moving	Key	Charger	Overall
Instance Space NN	130.8 ± 1.530	22.24 ± 0.441	7.764 ± 0.232	7.783 ± 0.214	42.13
Embedding Space LSTM	3.151 ± 0.662	13.04 ± 0.899	$\mathbf{0.360} \pm \mathbf{0.055}$	0.689 ± 0.124	4.311
Instance Space LSTM	7.313 ± 2.627	11.13 ± 1.169	0.488 ± 0.062	0.628 ± 0.126	4.890
CSC Instance Space LSTM	$\mathbf{0.605} \pm \mathbf{0.166}$	$\mathbf{5.307} \pm \mathbf{0.299}$	0.391 ± 0.083	$\mathbf{0.125} \pm \mathbf{0.012}$	**1.607**
Instance Space NN	0.217 ± 0.001	0.068 ± 0.000	0.011 ± 0.000	0.025 ± 0.000	0.080
Embedding Space LSTM	101.8 ± 60.35	3.033 ± 0.715	0.010 ± 0.008	0.037 ± 0.016	26.23
Instance Space LSTM	0.263 ± 0.038	0.069 ± 0.005	0.002 ± 0.000	0.005 ± 0.001	0.085
CSC Instance Space LSTM	$\mathbf{0.073} \pm \mathbf{0.016}$	$\mathbf{0.026} \pm \mathbf{0.002}$	$\mathbf{0.001} \pm \mathbf{0.000}$	$\mathbf{0.001} \pm \mathbf{0.000}$	**0.025**

CSC Instance Space LSTM approach leads to the best RL agent performance, coming closest to the oracle. Interestingly, for the Timer task, both the CSC Instance Space LSTM and the Instance Space LSTM actually outperform the use of the oracle, suggesting that the learnt hidden states are easier to exploit for policy learning than the raw timer state (we investigate what these models have learnt in Section 5.1). Note the very poor performance of agents trained without hidden state information, which aligns with expectations. For further details on agent training, see Appendix E.

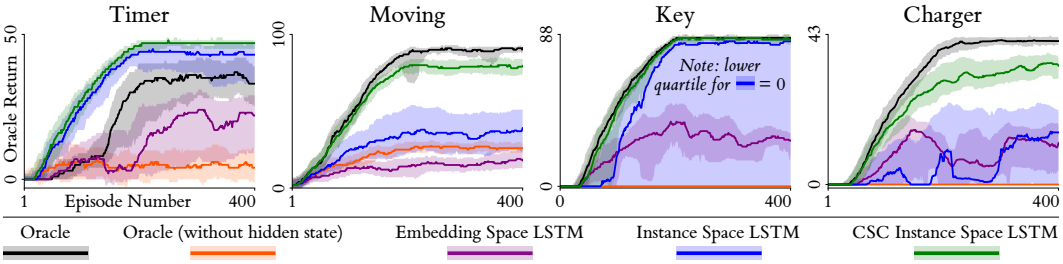

Figure 5: RL performance for different training configurations on our four RL tasks. The results given are the medians and interquartile ranges. For the oracle results, we trained ten repeats, and for the MIL-LSTM results, we performed one RL training run for each MIL-LSTM model repeat.

4.4 Robustness to Mislabelling

In this work, the return labels are provided by oracles rather than real people. When using human evaluators, there are likely to be imperfections and inconsistencies in the labels provided, and it is important to evaluate model robustness against such noise [24]. We implement noise through label swapping [30]; this ensures the marginal label distribution remains the same and does not include

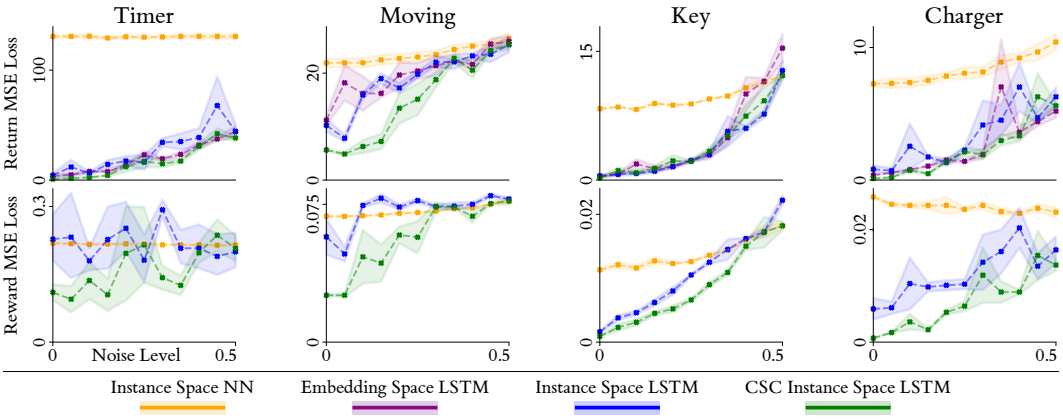

Figure 6: The return and reward performance of the MIL RM models (three repeats) subject to label noise. Note we omit the Embedding Space LSTM reward losses as they are very high.

out-of-distribution returns. In Figure 6, we show how both return and reward prediction decay with noise levels increasing from 0 (no labels swapped) to 0.5 (half swapped). The rate, smoothness, and consistency (across three repeats) of this decay varies between tasks, with decays in return prediction generally being smoother. We observe that the CSC Instance Space LSTM model remains the strongest predictor of both return and reward in the majority of cases, indicating a general robustness and providing evidence that the model should still be effective with imperfect human labels. On all metrics aside from Timer reward loss (where the mix of negative and positive rewards makes the effect of noise especially unpredictable), a noise level of at least 0.3 is required for the CSC Instance Space LSTM to perform as badly as the Instance Space NN baseline does with no noise at all.

5 Discussion

In this section, we seek to interpret our MIL RM models, analysing the distribution of learnt hidden states (Section 5.1) as well as their temporal dynamics over the course of a trajectory (Section 5.2). Finally, in Section 5.3 we discuss the limitations of this work and potential areas for future work.

5.1 Hidden State Analysis

The primary purpose of RM is to perform accurate reward reconstruction to facilitate agent training, but there is a secondary opportunity to improve understanding of human preferences through interpretability analysis of the learnt models. We can directly visualise the 2D LSTM hidden states of our oracle experiments, which enables a qualitative comparison of the interpretability of the various model architectures (see Figure 7). Visualising the hidden states with respect to the temporal dependencies indicates that the CSC Instance Space LSTM has learnt insightful hidden state representations. Breaking down the CSC Instance Space LSTM hidden embeddings: for the Timer task, time is represented along a curve, with a sparser representation around $t = 50$ (the crossover point when the treasure reward becomes positive). For the Moving task, time is similarly captured along with an additional notion of the change in treasure direction from right to left. For the Key task, the binary state of no key vs key is separated, with additional partitioning based on x position, denoting the two different start points of the agent. We discuss the Charger task in Section 5.2.

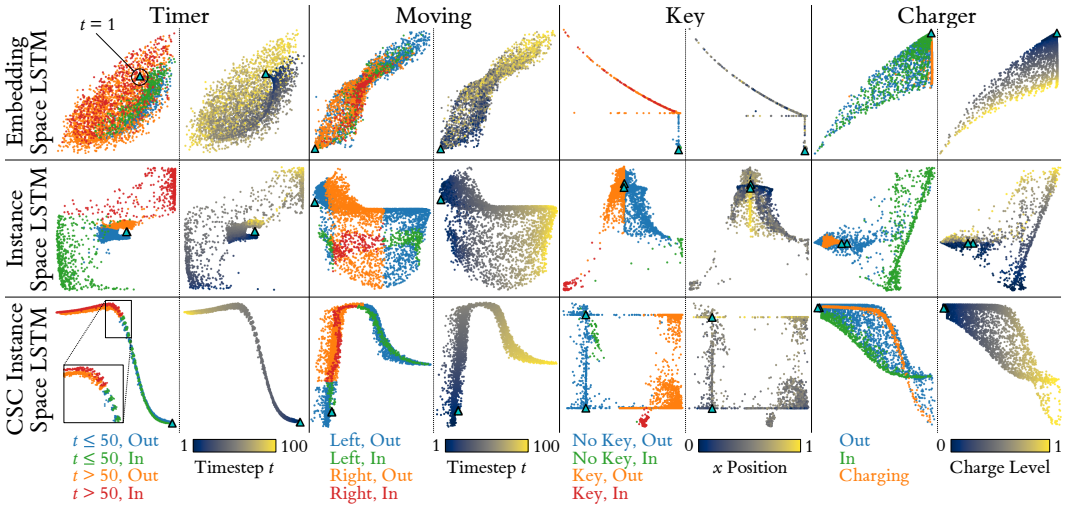

Figure 7: Learnt hidden state embeddings for our MIL RM models. For each model and task, we categorise the hidden state embeddings depending on the true environment state (first column for each task; In and Out indicate whether the agent is in the treasure or not; Left and Right indicate the direction in which the goal is currently moving). We also provide labelling based on temporal information (second column for each task). Furthermore, we include markers to indicate the hidden states for the centres of the agent spawn zones. For the Key task, as the temporal information is captured in the state categorisation (no key vs key), we use the second column to show the relationship between the hidden embeddings and the agent's x position. This helps to showcase the separation of the two possible start spaces learnt by the CSC Instance Space LSTM model. We elected to use the best performing repeat for each model as assessed by the reward reconstruction (see Table 1).

5.2 Trajectory Probing

We further interpret our models by visualising the learnt reward with respect to the environment state, and by using hand-specified *probe* trajectories to verify that the learnt hidden state transitions mimic the true transitions. We present the above for the CSC Instance Space LSTM model on the Charger task in Figure 8 (Appendix F contains similar figures for the other tasks). The top row shows that the model has correctly learnt the relationships between position, charge, and reward (reward increases only in the treasure zone as charge increases). From the probes, we can see how the charge level can be recovered from the hidden states. We also note that the inflection point between under-charging and over-charging is captured, i.e., this is where the optimal charge level lies, subject to some noise based on where the agent starts in the spawn zones. Furthermore, with the Challenging probe, we observe that the learnt hidden states align with the agent moving in and out of the charging zone.

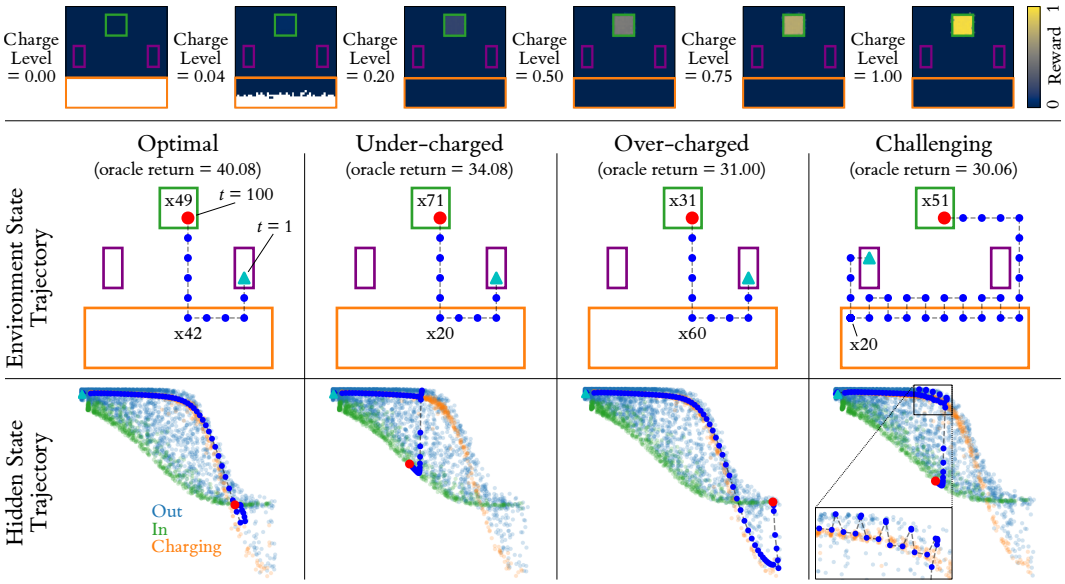

Figure 8: Interpretability for the CSC Instance Space LSTM on the Charger task. **Top**: the learnt relationship between agent position, charge level and reward. **Bottom**: Four probe trajectories demonstrating hidden state transitions as the trajectory progresses. *Optimal*: best possible return by charging to a sufficient level and maximising time in treasure. *Under-charged*: reward for treasure is too low. *Over-charged*: time is wasted charging after the maximum charge of 1 is reached. "xn" labels indicate the agent remaining in a position for n timesteps. As in Figure 7, this figure investigates the best performing model according to reward reconstruction.

5.3 Limitations and Future Work

Although we analyse the performance of our methods in the presence of noisy labels in Section 4.4, a major area of future work is to apply our methods to actual human data, potentially as a diagnostic tool to understand the extent to which real preferences are non-Markovian, and to generalise the learning to other information sources such as trajectory demonstrations [28] and pairwise choices [12]. Furthermore, we perform RM from an offline dataset before integrating fixed models into RL training; an iterative bootstrapping approach (learning from trajectories generated by the RL agent itself) is likely to be more practically useful. More generally, we hope that our identification of the link between RM and MIL may inspire a bidirectional transfer of tools and techniques.

6 Conclusion

We posed the problem of non-Markovian RM, which removes an unrealistic assumption about how humans evaluate temporally-extended agent behaviours. After identifying an isomorphism between RM and MIL, we proposed and evaluated novel MIL-inspired models that allow us to reconstruct non-Markovian reward functions, augment agent training, and interpret their learnt representations.

References

[1] D. Ariely and Z. Carmon. Gestalt characteristics of experiences: The defining features of summarized events. *Journal of Behavioral Decision Making*, 13(2):191–201, 2000.

[2] F. Bacchus, C. Boutilier, and A. Grove. Rewarding behaviors. In *Proceedings of the National Conference on Artificial Intelligence*, pages 1160–1167, 1996.

[3] F. Bacchus, C. Boutilier, and A. Grove. Structured solution methods for non-markovian decision processes. In *AAAI/IAAI*, pages 112–117. Citeseer, 1997.

[4] A. Bajcsy, D. P. Losey, M. K. O'Malley, and A. D. Dragan. Learning robot objectives from physical human interaction. In *Conference on Robot Learning*, pages 217–226. PMLR, 2017.

[5] B. Bakker. Reinforcement learning with Long Short-term Memory. *Advances in Neural Information Processing Systems*, 14, 2001.

[6] C. Basu, E. Bıyık, Z. He, M. Singhal, and D. Sadigh. Active learning of reward dynamics from hierarchical queries. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 120–127. IEEE, 2019.

[7] T. Bewley and F. Lecue. Interpretable Preference-based Reinforcement Learning with Tree-Structured Reward Functions. In *21st International Conference on Autonomous Agents and Multiagent Systems*, 2022.

[8] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym, 2016.

[9] M.-A. Carbonneau, V. Cheplygina, E. Granger, and G. Gagnon. Multiple instance learning: A survey of problem characteristics and applications. *Pattern Recognition*, 77:329–353, 2018.

[10] L. Chan, A. Critch, and A. Dragan. The impacts of known and unknown demonstrator irrationality on reward inference. 2020.

[11] L. Chan, A. Critch, and A. Dragan. Human irrationality: both bad and good for reward inference. *arXiv preprint arXiv:2111.06956*, 2021.

[12] P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei. Deep reinforcement learning from human preferences. *Advances in Neural Information Processing Systems*, 30, 2017.

[13] J. Early, C. Evers, and S. Ramchurn. Model Agnostic Interpretability for Multiple Instance Learning. In *International Conference on Learning Representations*, 2022.

[14] M. Hausknecht and P. Stone. Deep Recurrent Q-learning for Partially Observable MDPs. In *2015 AAAI Fall Symposium Series*, 2015.

[15] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[16] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[17] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[18] R. T. Icarte, T. Klassen, R. Valenzano, and S. McIlraith. Using reward machines for high-level task specification and decomposition in reinforcement learning. In *International Conference on Machine Learning*, pages 2107–2116. PMLR, 2018.

[19] M. Ilse, J. Tomczak, and M. Welling. Attention-based deep multiple instance learning. In *International conference on machine learning*, pages 2127–2136. PMLR, 2018.

[20] F. Jarboui and V. Perchet. Trajectory representation learning for multi-task nmrdp planning. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 6786–6793. IEEE, 2021.

[21] H. J. Jeon, S. Milli, and A. Dragan. Reward-rational (implicit) choice: A unifying formalism for reward learning. *Advances in Neural Information Processing Systems*, 33:4415–4426, 2020.

[22] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

[23] D. Kahneman. Evaluation by moments: Past and future. *Choices, values, and frames*, pages 693–708, 2000.

[24] K. Lee, L. Smith, A. Dragan, and P. Abbeel. B-pref: Benchmarking preference-based reinforcement learning. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2021.

[25] J. Leike, D. Krueger, T. Everitt, M. Martic, V. Maini, and S. Legg. Scalable agent alignment via reward modeling: a research direction. *arXiv preprint arXiv:1811.07871*, 2018.

[26] D. Li and Y. Zhang. Multi-instance learning algorithm based on lstm for chinese painting image classification. *IEEE Access*, 8:179336–179345, 2020.

[27] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[28] A. Y. Ng, S. J. Russell, et al. Algorithms for inverse reinforcement learning. In *International Conference on Machine Learning*, volume 1, page 2. PMLR, 2000.

[29] S. Reddy, A. Dragan, S. Levine, S. Legg, and J. Leike. Learning human objectives by evaluating hypothetical behavior. In *International Conference on Machine Learning*, pages 8020–8029. PMLR, 2020.

[30] D. Rolnick, A. Veit, S. Belongie, and N. Shavit. Deep learning is robust to massive label noise. *arXiv preprint arXiv:1705.10694*, 2017.

[31] S. Russell. *Human Compatible: Artificial Intelligence and the Problem of Control*. Penguin, 2019.

[32] A. Shah, S. Wadhwania, and J. Shah. Interactive robot training for non-markov tasks. *arXiv preprint arXiv:2003.02232*, 2020.

[33] R. Shah, N. Gundotra, P. Abbeel, and A. Dragan. On the feasibility of learning, rather than assuming, human biases for reward inference. In *International Conference on Machine Learning*, pages 5670–5679. PMLR, 2019.

[34] Z. Shao, H. Bian, Y. Chen, Y. Wang, J. Zhang, X. Ji, et al. Transmil: Transformer based correlated multiple instance learning for whole slide image classification. *Advances in Neural Information Processing Systems*, 34, 2021.

[35] S. Thiébaux, C. Gretton, J. Slaney, D. Price, and F. Kabanza. Decision-theoretic planning with non-markovian rewards. *Journal of Artificial Intelligence Research*, 25:17–74, 2006.

[36] R. Toro Icarte, E. Waldie, T. Klassen, R. Valenzano, M. Castro, and S. McIlraith. Learning reward machines for partially observable reinforcement learning. *Advances in neural information processing systems*, 32, 2019.

[37] M. Tu, J. Huang, X. He, and B. Zhou. Multiple instance learning with graph neural networks. *arXiv preprint arXiv:1906.04881*, 2019.

[38] H. Van Hasselt, A. Guez, and D. Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

[39] K. Wang, J. Oramas, and T. Tuytelaars. In defense of lstms for addressing multiple instance learning problems. In *Proceedings of the Asian Conference on Computer Vision*, 2020.

[40] X. Wang, Y. Yan, P. Tang, X. Bai, and W. Liu. Revisiting multiple instance neural networks. *Pattern Recognition*, 74:15–24, 2018.

[41] D. Wierstra, A. Förster, J. Peters, and J. Schmidhuber. Recurrent policy gradients. *Logic Journal of the IGPL*, 18(5):620–634, 2010.

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? [Yes] See Section 5.3
   (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)? [Yes] See Appendix A
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendices C, D, and E
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Appendix 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)? [Yes] See Appendix A
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? [N/A]
   (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? [Yes] Our source code and models, which are included in the supplementary material and will be made available publicly upon acceptance.
   (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]