Active physical inference via reinforcement learning

Abstract

When encountering unfamiliar physical objects, children and adults often perform structured interrogatory actions such as grasping and prodding, so revealing latent physical properties such as masses and textures. However, the processes driving and supporting these curious behaviors are still largely mysterious. In this paper, we develop and train an agent able to actively uncover latent physical properties such as the mass and force of objects in a simulated physical "micro-world'. Concretely, we used a simulation-based-inference framework to quantify the physical information produced by observation and interaction with the evolving dynamic environment. We used reinforcement learning to train an agent to implement general strategies for revealing latent physical properties. We compare the behaviors of this agent to the human behaviors observed in a similar task.

Keywords: physical simulation; active learning; probabilistic inference; reinforcement learning

Human adults have an intuitive understanding of the physical world that supports rapid and accurate predictions, judgments and goal-directed actions. For example, we can quickly estimate how heavy a door is by how it responds to a push, judge whether a tower is stable with a glance, or predict where to stand to catch a baseball by watching its trajectory. These abilities are so ingrained in everyday life, we are rarely aware of the complexity of identifying physical properties of objects from brief experiences and interactions with their dynamics.

Bramley, Gerstenberg, Tenenbaum, and Gureckis (2018) explored the strategies people use to actively infer masses and forces of attraction and repulsion that relate objects in a simple simulated environment. Their learning environment consisted of a bounded two-dimensional "hockey puck" world, containing four circular objects of varying masses and related with pairwise attractive or repulsive magnet-like forces. For subjects, the world was displayed on a computer screen and updated in real time using a physics engine that approximates Newton's laws of motion (see Figure 1a). Subjects could hold-click to "grab" onto any object then move their cursor to "drag" that object around the scene, so altering how the scene played out and often better revealing the physical property they had been tasked with inferring. Bramley et al. (2018) used a simulation-based inference model to assess what information was generated by subjects, and contrasted trials in which the subject's goal was to identify the relative masses of two objects against trials in which their goal was to identify the pairwise force between two objects. They found that learners were able to gather information selectively relevant to their learning goal, and exhibited markedly different behaviours dependent on the goal that could be classified into a classes of frequent experimental strategies such as staging collisions by *throwing* objects at one another, *shaking* them back and forth, bringing pairs of objects close together (dubbed *encroaching*). However, they did not model how such strategies were learned or generalised successfully across learning instances

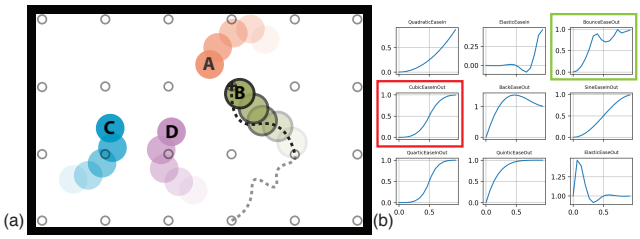

Figure 1: Visualisation of task environment adapted from Bramley et al. (2018). Gray circle overlay shows grid of target locations and dashed line give an example control trajectory in which B is grabbed and moved toward A. (b) Sample of the interpolation functions used to create smooth control trajectories.

In the current work, we take a step toward understanding this ability. We use reinforcement learning to train an artificial agent to minimize its own uncertainty about these specific physical parameters by composing action sequences from an action space qualitatively similar to human subjects' mouse movements, training this ability across a diverse set of ground truth and initial conditions. We are interested whether, with sufficient training and subjective information as a reward signal, an agent can learn to produce robustly informative action sequences, or strategies, for revealing particular physical properties, and whether these will reflect the behaviors previously seen in human subjects.

Related Work

Understanding what drives information-seeking behavior has long been a goal in psychology (see Schulz & Gershman, 2019, for a recent review). Some approaches attempt to tie information seeking directly (e.g., Guez, Silver, & Dayan, 2012) or incidentally (e.g., Thompson, 1933) to the familiar goal of extrinsic reward maximisation. Explicit approaches require intensive preposterior planning, that is averaging behavioral prescriptions over many potential future learning and reward trajectories (Raiffa, 1974), so are infeasible for complex problems. Incidental approaches are computationally cheaper but fail to capture information seeking behaviours that depart from noisy extrinsic goal seeking (cf., Schulz, Klenske, Bramley, & Speekenbrink, 2017).

The idea that the brain seeks information as a form of *intrinsic* reward captures a middle ground idea that that for humans, discovery is an "end in itself" driven by something we intuitively understand as "curiosity" (Gottlieb, Oudeyer, Lopes, & Baranes, 2013; Schmidhuber, 2010).

While some learning problems only occur once in a life-

time (e.g., we only need to figure out the laws of physics once), many others occur repeatedly (we encounter new objects daily and would like to be able to rapidly infer their most important properties). For these problems, there is space to *learn to actively learn* by training reusable active learning strategies through repeated experience,making them amenable to reinforcement learning (Kober & Peters, 2012).

The task we explore is challenging in part because of the continuous and complex nature of physical dynamics. However, combined with adequate function approximators such as a deep neural network, reinforcement learning has proven successful for optimising control in rich state spaces (Mnih, Heess, Graves, et al., 2014). For example, Bachman, Sordoni, and Trischler (2016) developed agents that can solve a collection of tasks which require active information seeking using deep neural networks and reinforcement learning, including cluttered MNIST (Mnih et al., 2014), Blockworld, CelebA (Liu, Luo, Wang, & Tang, 2015) and Hangman.

This project sets apart from prior work on intrinsic curiosity in that it focuses on interventional rather than observational information seeking. In our physics learning task, actions have extended complex and far reaching consequences, compounding the complexity of inference, but better mimicking the challenges of learning in the natural world.

The task

Our interactive physical environment adapts the two-dimensional physics-based environment from Bramley et al. (2018) and is implemented using the `pybox2d` library.[1] The world is limited to a 6×4 meter bounded rectangle containing four circular objects, each with radius 0.25 meters. The objects interact with one another and with the static walls according to Newtonian physics and the latent properties of mass and the pairwise forces. The complete fixed settings of the simulator are detailed at `https://bit.ly/2B4TOAf`.

Possible settings and initial uncertainty

As with Bramley et al. (2018), we will contrast actions focused on identifying objects' *masses* against actions focused on identifying the *forces* relating each pair of objects. However, we explored a larger set of possible settings of these values and correspondingly larger initial hypothesis space for the agent than human participants. To ensure the agent had equivalent initial uncertainty about both masses and forces, we defined a discrete space of 2^5 mass combinations and 2^5 force combinations and initialised the agent with a uniform prior across all 2^{10} distinct mass-force combinations. The space of mass combinations is a subset of $\mathbb{R}^4$, in which each parameter represents the mass of an object and is in the range of [1, 3]kg. Meanwhile, the space of force combinations is a subset of $\mathbb{R}^{4\times4}$, in which each force parameter takes a value $\in (-3,0,3)^{\mathrm{m}/\mathrm{s}^2}$, representing repulsive force, no interactive force, and attractive force respectively for each pair of objects. Under the Newton's second law of motion, different mass-force combinations yield different accelerations, leading to distinct simulated trajectories, interactions and collisions.

Like the human participants in Bramley et al. (2018), the agent could "grab" one object at a time. The cursor would exert an elastic force (i.e., scaling continuously with the inverse square of the distance between the object and the cursor) attracting the controlled object to the cursor's location until it was released. All interactions and resulting trajectories were simulated by the Box2D engine.

The Agent

Learning Framework

Reinforcement learning (RL) focuses on how agents learn sequential control policies to maximize cumulative reward. We assume (S,A,R,T) defines a Markov Decision Processes (MDP) with state space S, action space A, reward function R and transition dynamics T. Modern RL can be divided into model-based and model-free approaches, where the former first learns a predictive a world model then uses the model to learn a policy, while the latter learns an optimal policy directly from experience. We position our learning agent somewhat like a developing child learning about the physical properties of the world through experience but without much *a priori* knowledge. Thus, we do not assume the agent has a complete internal physics engine from which to draw samples (model-based learning) but instead is learning how best to reveal information in a model-free manner evaluated against its own learning progress (Oudeyer, Kaplan, & Hafner, 2007).

Action space

As mentioned above, human subjects exhibited rich and informative behaviors when interacting with the objects in this environment. However, many of the strategies identified in Bramley et al. (2018) appeared to be composed of multiple simpler movements. For instance, "shaking" involved grabbing then moving an object rapidly back and forth between two locations while "throwing" involved grabbing an object and releasing it at speed and in a direction such that it went on to collide with another. To encode an action space expressive enough to incorporate these realistic and extended human behaviors, we combined a a grid of target locations (Figure 1a) with pool of easing (interpolation) functions (see Figure 1b). Concretely for each action, the agent chose an object to control (none, 1,2,3 or 4) and a cardinal direction (up, down, left ,right), or a quadrant (up-right, up-left, down-left, down-right) which determined a target location adjacent to its current location. It then followed a path to that location determined by its selected interpolation function. Figure 1b shows these functions in the first quadrant such that the path would be appropriately mirrored in the other three quadrants.[2] Together with horizontal and vertical movements, and no movement — i.e., the cursor pausing at its current position — our

[1] `https://github.com/pybox2d/pybox2d`

[2] All together there were (stay + $\rightarrow$ + $\uparrow$ + $\leftarrow$ + $\downarrow$ + 31[functions] × 4[quadrants]) × (4[objects] + no object)=645 possible actions.

action generator output a contiguous mouse trajectory, following the policy learned by our agent via the learning framework demonstrated in the following section.

A complete learning episode consisted of a sequence of these actions and would terminate when the agents' uncertainty about a target property of the environment reached a threshold, or a timeout limit was reached. For simplicity we assumed each action occurred across a fixed time window, and to ensure continuity of mouse movement, the start point of each action was the end position for the last action. This resulted in a continuous action trajectory decomposed into a sequence of discrete action choices.

State definition

We defined the motion of an object at every time step t with scalars m_t and ϕ_t, where m_t represents the magnitude and ϕ_t the direction of the object. Given a two-dimensional space, $m_t = \sqrt{v_{x_t}^2 + v_{y_t}^2}$ and $\phi_t = \arctan \frac{v_{y_t}}{v_{x_t}}$, where $v_t = [v_{x_t}, v_{y_t}]$ is the velocity vector of the object at time step t. Together with the location tuple (x_t, y_t), the object state $s_t \in S$ is thereby a four-dimensional vector $[m_t, \phi_t, x_t, y_t]$.

Intrinsic Reward Signal

Our agents' goal was to minimise its final uncertainty about either mass or force. Thus its reward signal was based on computing its reduction in entropy with respect to the target property (Shannon, 1951). Following Bramley et al. (2018), we assumed likelihoods were computed based on divergence between mental simulations and the observed trajectories.

Concretely, to infer how likely a mass-force combination, w, is we assume actual observed dynamics are compared against dynamics simulated assuming those properties. Let $w \in W$, where W is the space of all possible mass-force combinations. Before an episode starts, the agent has a uniform prior over settings $p(W)$. After a period of action and observation d, we assess the likelihood of the observed trajectory o under all possible w, and use this to update the prior distribution $p(W|d)$.

Following Vul, Frank, Alvarez, and Tenenbaum (2009), we modelled the likelihood of observing the object trajectories o given the potential property setting w and the mouse trajectory a using a Gaussian error distribution:

$$p(o \mid w, a, \beta) = \prod_{t=1}^{T} \exp^{-\frac{\beta}{2\Sigma}(s_t - d_t)^\top (s_t - d_t)}, \tag{1}$$

where d_t is the observed $[m_t, \phi_t]$ produced by the true environment w'. The covariance matrix was $\Sigma = \begin{bmatrix} \sigma_m^2 & 0 \\ 0 & \sigma_\phi^2 \end{bmatrix}$ where σ_m^2 and σ_ϕ^2 were set to the empirical standard deviations of the disparities between simulations and the actual observations in Bramley et al. (2018). β was a scaling parameter determining how confidently the agent could perceive divergences between the objects' true and simulated trajectories.[3]

[3] Intuitively, $\beta = 0$ would result in no learning and $\beta \lim \infty$ would lead to an implausibly powerful learner with infinite perceptual precision. In our experiment, we assumed a constant β of $\frac{1}{50}$. We also held a fixed size of the time window T of $\frac{1}{6}$ seconds, over which forward simulations were compared against observations before being corrected.

According to Bayes' rule, once we measure the likelihood through Equation 1, we can calculate the posterior of the potential latent physics properties w by:

$$p(w \mid o, a, \beta) = \frac{1}{Z} p(o \mid w, a, \beta) p(w), \tag{2}$$

where $p(w)$ is the prior and Z is a normalizing constant. Using Shannon entropy (1951) as a measurement of the amount of remaining uncertainty about w', we formulated the immediate reward after action t as:

$$r_t = -\Big(\sum_{w \in W} p(w) \log p(w) - \sum_{w \in W} p(w \mid o, a, \beta) \log p(w \mid o, a, \beta)\Big). \tag{3}$$

For particular parameters of interest (here masses of the four objects), the posteriors and priors are marginalized over the remaining parameters (in this case, the pairwise forces) when estimating the parameter specific uncertainty reduction. The resultant reward signal r embodies the amount of information about the latent physics parameters (mass or force) a particular agent's action has obtained.

Deep Q-Learning

Q-learning is a model-free learning algorithm that estimates the expected cumulative discounted rewards of performing an action from a given state (Watkins & Dayan, 1992). These estimated cumulative discounted rewards are often called returns and represented as Q-values (state-action values). One way to iteratively learn Q-values is by using a *Temporal Difference* (TD) algorithm (Sutton, 1988). TD updates Q-values by:

$$\hat{Q}(s,a) := (1-\alpha) Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s', a')), \tag{4}$$

where (s,a) is the current state-action pair, (s',a') is the state-action pair at the next time step, γ is the discounted factor, and α is a learning rate.

A challenge for conventional Q-learning is to estimate Q-values over continuous states and action spaces. For our task, despite creating a discrete action space with interpolation functions, the object's locations and motion $[m, \phi]$ was continuous throughout the world. Computing separate Q-values for every possible (s,a) pair at every time window is infeasible and naïve given the smooth relationships between nearby states. We hence adapted the *deep* Q-learning method, which instead of updating Q-values of each discrete pair (s,a) in tabular form, approximates $\hat{q}(s,a)$ with deep neural network. Denoting the parameters of the network as θ, a mean squared error loss of the Q-values approximator can be written as:

$$l(s, a \mid \theta) = \frac{1}{2}(r + \gamma \max_{a'} \hat{q}(s', a' \mid \theta) - \hat{q}(s, a \mid \theta))^2, \tag{5}$$

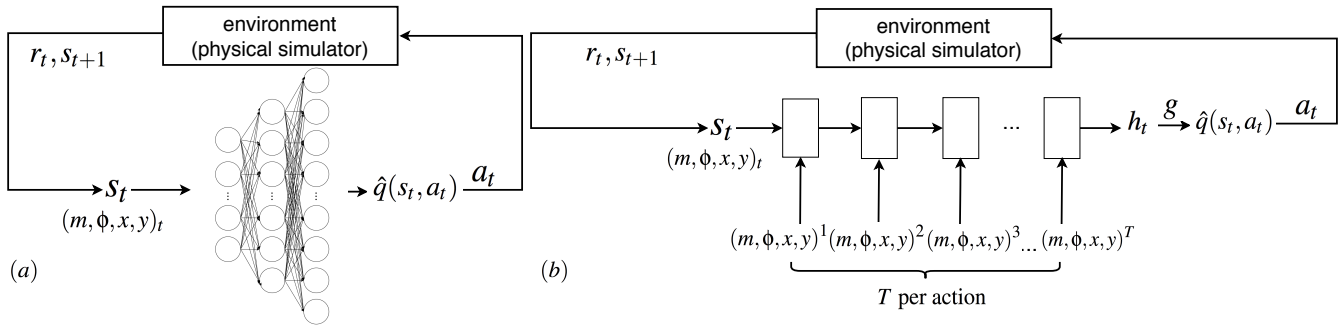

Figure 2: Proposed (a) MLP and (b) RQN structures of our Q-learning framework. For every time period, the network takes a sequence of object states and return the estimated Q-values. The control policy of the next time period, a sequence of mouse trajectories, is determined by selecting the action corresponding with the largest Q-values over the action space.

where $r+\gamma\max_{a'}\hat{q}(s',a' \mid \theta)$ is the *target* of our estimation. Below, we present two neural network models, serving as the function approximators $\hat{q}_\theta$. Optimal control strategies can be discovered following this general learning structure.

Multilayer Perceptron First, we used a Multi-Layer Perceptron (MLP) with three layers to approximate Q-values (Figure 2a). For every time period t, the simulator fed the agent a sequence of object states s_t that spanned over T time-steps and offered a reward r_t to the MLP. The network outputs a deterministic cursor trajectory that was the action $a_t \in A$ with largest Q-value at t. We used ε-greedy exploration and optimized Equation 5 via stochastic semi-gradient descent (Watkins & Dayan, 1992). Note that the same network $\hat{q}$ producing the next state target Q-values was used for computing the loss of the current predictions. Such optimization can yield erratic gradients when the control policy oscillates. To deal with this potential instability, we redefined the *target* as a *target network*. Instead of using the same network to compute the next state target and the current state Q-values, another network $q_{\hat{\theta}^-}$, sharing the same structure as $\hat{q}_\theta$, was utilized to compute target Q-values during the update. The loss function in Equation 5 was then modified as:

$$l(s,a \mid \theta,\theta^-) = \frac{1}{2}\left[r+\gamma\max_{a'}\hat{q}(s',a' \mid \theta^-) - \hat{q}(s,a \mid \theta)\right]^2. \quad (6)$$

Following the 'hard-copy' update proposed in Mnih et al. (2015), we froze θ^- and copied θ into θ^- once after a few episodes. Experiments demonstrated that the MLP with *target network* model produced convergent and stable policies faster than the simple MLP model.

Recurrent Q-Network For every forward pass, the function approximator $\hat{q}_\theta$ should receive a sequence of object states with length $16T$ (four objects, each carrying a four-dimensional vector $[m,\phi,x,y]$ that depicts the object's motion per time step). An MLP approximator, as introduced in the previous section, consists of multiple fully-connected layers, where each layer contains multiple neurons that accept and send information across the network. The input motion vectors are stacked and reshaped into a $16T$-dimensional vector, and then fed into the first layer of the MLP. However, the state vectors encode the objects' movement information over a period of time, and simply flattening them may fail to capture some parts of the underlying dynamics or spatial correlations among the objects. We thereby propose a second RNN-based function approximator (Figure 2b).

Instead of treating all the motion vectors equally without order, RNN chronologically receives the motion vectors and updates its hidden cells accordingly. The output hidden vector at the last time step included not only the object state information but also their latent interconnections with the environment. To alleviate the vanishing gradient issue, we adapted the Long Short-Term Memory cell (Hochreiter & Schmidhuber, 1997) as the recurrent unit. A linear mapping function $g : S \rightarrow A$ mapped the state representation to actions. We optimize this recurrent Q-network (RQN) by semi-gradient descent using the *target network* method.

Experiments

In this section, we present some preliminary attempts at training these networks and comparing their behavior to that of humans performing the same task. We compare the achieved reward with respect to the agent's goal against its reward for the mismatched alternative goal to assess to what extent the actions were selectively informative.

Training

For our physics simulator, 60 time-steps was equivalent to 1 second. We set the time-window T as 40 time-steps, implying that each atomic action would take roughly 0.7 seconds. It is nontrivial to determine when a simulated game should stop, since uncertainty typically continues to diminish indefinitely, approaching but never reaching zero. For our agent, the total uncertainty of the latent physical parameter of interest could be obtained by computing the Shannon entropy of the initial prior distribution $p(W)_0$, denoted $\mathrm{H}(p(W)_0)$. We denoted a reward threshold factor $\gamma_r = 0.95$. Then, every time the cumulative reward reached $\gamma_r \mathrm{H}(p(W)_0)$, the game would stop. Otherwise, the agent would continue playing until the current game approached a timeout limit.

The three-layer MLP had 150, 250, and 450 neurons per layer. The input state s_t of MLP was in R^{640}, while for RQN, it was R^{16x40}. The approximated Q-values $\hat{q}_\theta(s,a) \in R^{645}$. We initialized the exploration rate ε as 0.5, discounted by a factor

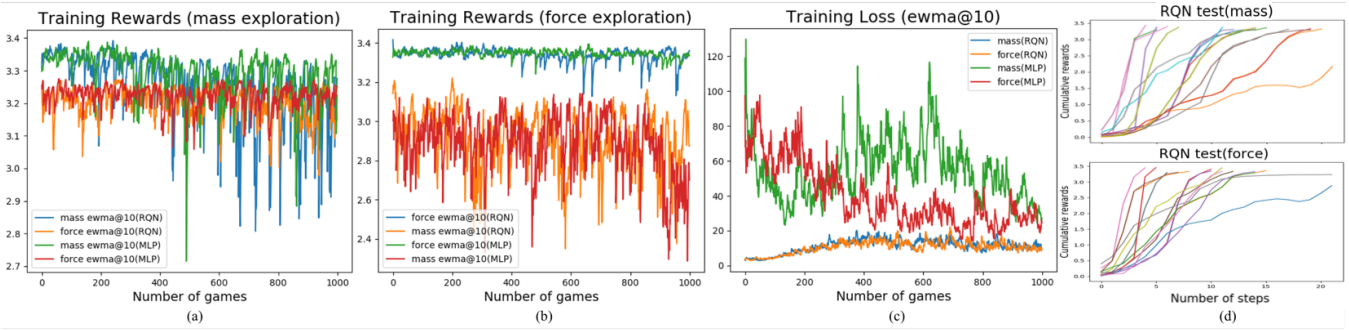

Figure 3: Reward and loss. (a) and (b) show total target and mismatched reward for each game, for the mass and force exploration tasks respectively. (c) reports the training loss associated with all proposed models. For each test game, the cumulative target rewards over actions are recorded and presented in (d).

of 0.95 every 20 games until ε reached 0.01. The discount factor γ was 0.99, and the weights of *target network* were cloned from $\hat{q}_\theta$ every 20 games.

A training set with 60 distinct ground truth w, initial object locations, velocities, cursor positions, and initial velocities was created, and a holdout test set containing 20 different configurations, are defined beforehand to ensure the robustness of the models. For each proposed model, we trained the agent for the mass exploration task and the force exploration task separately with 1000 episodes, and the training errors and rewards are illustrated in Figure 3(a)-(c). Within each task, both the target reward and the mismatched reward (i.e., force in a mass exploration task) are recorded. To better illustrate the moving trends, we weighted the training rewards and the loss by exponentially weighted moving average (ewma) with span 10.

Compared with the MLP-based models, RQN converged faster with small and stable loss, as shown in Figure 3(c). Both models produced swinging errors in the training process. The intuition is that 'hard' copying weights from θ into θ^- yielded a time interval when θ^- was frozen and the predicted Q-values diverged from the target Q-values. Such oscillations would not distort the optimization of the objective function, as the time interval was small and θ^- was constantly updated, ensuring steady amounts of information explored by the agent. Given a flexible timeout limit, both methods were able to uncover the uncertainty of the environment, as depicted in Figure 3(a) and (b). Mass reward was more variable. This is in line with the Bramley et al. (2018) finding that evidence about *mass* tends to come in sporadic spikes when objects collide or are moved rapidly, while *force* information typically accumulate more smoothly whenever objects are in close proximity. Thus, gathering mass information reliably have depended on more specific and targeted actions while moving objects closer together may have been sufficient for force information. Overall, the RQN exhibited continuing amounts of achieved information on all latent physics parameters. We applied the trained RQN agent on the test configuration set for mass and force exploration separately, each with 20 games. As reported in Figure 3(d), in most of the cases the agent could effectively capture the underlying properties of the environment within 20 steps. Since explorations were excluded from the testing, few 'abnormal' actions appeared. For those rare cases, our agent quickly came back to the right track and could still complete the learning task within the timeout limit.

Using the trained models we created a small dataset of 10 *force*-focused episodes and 10 *mass*-focused episodes using a holdout set of new worlds and starting locations. See `https://bit.ly/2FYTjvD` for videos. Comparing these episodes, we found that achieved information reward was similar for mass or force focused trials 2.82 ± 0.54, 2.96 ± 0.39, $t(18) = .98$, $p = .32$ but that significantly more information was generated for the property matched with the agents' goal than the alternative property (see also lower orange and blue lines in Figure 3a and b). That is, the agent generated more evidence and reward on average about the target property 3.1 ± 0.35bits than the mismatched property 2.7 ± 0.52 bits $t(18) = 2.4, p = 0.021$ similarly to human subjects in Bramley et al. (2018).

Trained agent behaviour

The agent frequently moved the controlled object closer to the other objects, reducing the distance to the closest object from 1.28 ± 0.24m to 1.01 ± 0.23m on average during each control action $t(38) = 3.6, p < .001$. As we see in Figure 4a, the agent learned that moving was normally more informative than staying still. There were also hints of goal dependent control strategies similar related to those identified in Bramley et al. (2018). For example, the most frequently selected interpolation by the mass focused agent was *bounceEaseOut* (Figure 4b and green highlight in Figure 1b), a particularly dynamic motion consistent with the rapid changes of direction associated with shaking or knocking observed frequently in the human data on *mass* focused trials. The intuition both there and here is that these actions strongly reveal objects mass by causing rapid changes in objects' directions. Meanwhile, the most frequent interpolation selected by force focused agent was *cubicEaseInOut* (Figure 4c and red highlight Figure 1b), a smooth motion intuitively consistent with the "encroaching" behavior observed frequently in force trials in Bramley et al. (2018) and effective in providing strong evidence about mass.

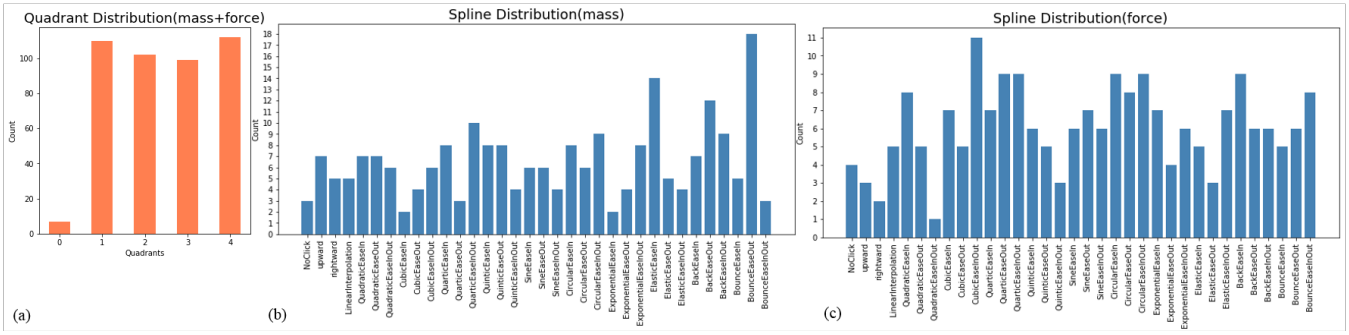

Figure 4: Distributions of emerged actions in 20 test games (10 for mass focused and 10 for force focused). (a) Sum of the quadrant distribution of mass and force tasks. '0' means the agent staying over at its position within the time-windows. (b) Spline distribution of mass task (c) Spline distribution of force task

Discussion and Conclusions

Humans display sophisticated intervention strategies when actively inferring the properties of physical objects. We used model-free reinforcement learning, deep function approximation, and simulation based inference to build an agent able to efficiently reveal the latent physical properties in human-like ways without external input. Part of the insight gleaned from this project comes from our solutions to the engineering challenges involved in creating a successful agent. To produce extended actions with richness and qualitative correspondence with humans', we found success with an action space that combined a discrete set of target locations with a discrete set of smoothing splines. We found that learning to associate action sequences with successful resolution of uncertainty was much more effective with a recurrent network architecture, but that robust strategies could be learned through model-free Q-learning. Following the predicted optimal control policies, not only did the agent uncover the latent parameters of interest, but there were also hints of behavioral correspondence with human subjects in that our mass trained agent would select more jagged and dynamic trajectories aligned with the strategies observed in Bramley et al. (2018).

While this study provides a valuable first step to understanding how humans learn and apply rich interrogatory behaviours when interacting with the natural world, it also has its limitations. One of these is the use of the physics simulator to calculate the reward signal. This is a rational idealisation of physics inference, but embodies the overly strong assumption that the agent is able to simulate the world accurately and perform approximate Bayesian inference with its own interaction data. A more plausible and practically viable approach to rewarding informative control would be to train a separate prediction network to anticipate upcoming dynamics, and use some function of its loss over time as a reward signal (cf. Oudeyer et al., 2007; Pathak, Agrawal, Efros, & Darrell, 2017). A more integrated agent could also use the predictor network approximation to plan actions that are likely to be informative through preplay Chentanez, Barto, and Singh (2005). Following Haber, Mrowca, Fei-Fei, and Yamins (2018), a complex adversarial-based learning framework may be helpful. In future work we plan to combine more realistic intrinsic rewards, richer action space and model based planning to better mimic the ability of humans to create intuitive physical experiments.

References

Bachman, P., Sordoni, A., & Trischler, A. (2016). Towards information-seeking agents. *arXiv preprint arXiv:1612.02605*.

Bramley, N. R., Gerstenberg, T., Tenenbaum, J. B., & Gureckis, T. M. (2018). Intuitive experimentation in the physical world. *Cognitive Psychology*, *105*, 9 - 38.

Chentanez, N., Barto, A. G., & Singh, S. P. (2005). Intrinsically motivated reinforcement learning. In *Advances in neural information processing systems* (pp. 1281–1288).

Gottlieb, J., Oudeyer, P.-Y., Lopes, M., & Baranes, A. (2013). Information-seeking, curiosity, and attention: computational and neural mechanisms. *Trends in cognitive sciences*, *17*(11), 585–593.

Guez, A., Silver, D., & Dayan, P. (2012). Efficient bayes-adaptive reinforcement learning using sample-based search. In *Advances in neural information processing systems* (pp. 1025–1033).

Haber, N., Mrowca, D., Fei-Fei, L., & Yamins, D. L. (2018). Learning to play with intrinsically-motivated self-aware agents. *arXiv preprint arXiv:1802.07442*.

Hochreiter, S., & Schmidhuber, J. (1997, November). Long short-term memory. *Neural Comput.*, *9*(8), 1735–1780.

Kober, J., & Peters, J. (2012). Reinforcement learning in robotics: A survey. In *Reinforcement learning* (pp. 579–610). Springer.

Liu, Z., Luo, P., Wang, X., & Tang, X. (2015). Deep learning face attributes in the wild. In *Proceedings of the ieee international conference on computer vision* (pp. 3730–3738).

Mnih, V., Heess, N., Graves, A., et al. (2014). Recurrent models of visual attention. In *Advances in neural information processing systems* (pp. 2204–2212).

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*, 529–533.

Oudeyer, P.-Y., Kaplan, F., & Hafner, V. V. (2007). Intrinsic motivation systems for autonomous mental development. *IEEE transactions on evolutionary computation*, *11*(2), 265–286.

Pathak, D., Agrawal, P., Efros, A. A., & Darrell, T. (2017). Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning (icml)* (Vol. 2017).

Raiffa, H. (1974). *Applied statistical decision theory*. Wiley.

Schmidhuber, J. (2010). Formal theory of creativity, fun, and intrinsic motivation (1990–2010). *IEEE Transactions on Autonomous Mental Development*, *2*(3), 230–247.

Schulz, E., & Gershman, S. J. (2019). The algorithmic architecture of exploration in the human brain. *Current opinion in neurobiology*, *55*, 7–14.

Schulz, E., Klenske, E., Bramley, N., & Speekenbrink, M. (2017). Strategic exploration in human adaptive control. *bioRxiv*, 110486.

Shannon, C. E. (1951, 01). Prediction and entropy of printed english. *Bell System Technical Journal*, *30*, 50-64.

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine Learning*, *3*, 9-44.

Thompson, W. R. (1933). On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, *25*(3/4), 285–294.

Vul, E., Frank, M. C., Alvarez, G. A., & Tenenbaum, J. B. (2009). Explaining human multiple object tracking as resource-constrained approximate inference in a dynamic probabilistic model. In *Advances in neural information processing systems* (p. 1955-1963).

Watkins, C. J., & Dayan, P. (1992). Q-learning. *Machine learning*, *8*(3-4), 279–292.