Goal-Induced Inverse Reinforcement Learning

Katie Luo

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2019-81
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2019/EECS-2019-81.html

May 17, 2019

Acknowledgement

I would like to thank my research advisor Professor Sergey Levine for the opportunity to work in RAIL Lab and providing me the resources and opportunities for research during my time in UC Berkeley. I would also like to thank Professor Pieter Abbeel for getting me into machine learning. I would especially like to thank my mentor, Justin Fu for the guidance and support in all my research endeavours. Last, but not least, thank you to everyone on the 4th floor of Sutardja Dai Hall for the support and humor that decreased my research productivity as well as the tea from 7th floor Sutardja Dai Hall which has provided me with the caffeine that restored it.

And as always, to my family.

Goal-Induced Inverse Reinforcement Learning

by Katie Z Luo

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

Committee

Sergey Levine

Professor Sergey Levine
Research Advisor

5/16/2019

(Date)

⋆ ⋆ ⋆ ⋆ ⋆ ⋆ ⋆

[illegible]

Professor Pieter Abbeel
Second Reader

5/17/19

(Date)

Goal-Induced Inverse Reinforcement Learning

Abstract

Goal-Induced Inverse Reinforcement Learning

by

Katie Z Luo

Master of Science in Electrical Engineering and Computer Science

University of California, Berkeley

,

Inverse reinforcement learning holds the promise for automated reward acquisition from demonstrations, but the rewards learn generally cannot transfer between tasks. We propose a framework that is able to learn a reward function that is language-aware on a multi-task setting.

This work presents Goal-Induced Inverse Reinforcement Learning, an IRL framework that learns a transferable reward function and achieves good performance as compared to imitation-learning algorithms. By learning the rewards in the IRL framework, our algorithm is able to obtain a more generalizable reward function that is able to solve different tasks by changing just the goal specification. Indeed, this work showed that the reward function learned changes to match the task at hand, and can be toggled depending on the given goal-instruction, mapping to the true, underlying reward function that the goal-instruction intends. This work also shows that the learned reward is shaped, allowing for ease learning by reinforcement learning agents. Furthermore, by training the policy and reward models jointly, we are able to efficiently obtain a policy that can perform on par with other imitation-learning policies. GIIRL shows comparable, if not better, results than behavioral-cloning algorithm.

Contents

Acknowledgments

I would like to thank my research advisor Professor Sergey Levine for the opportunity to work in RAIL Lab and providing me the resources and opportunities for research during my time in UC Berkeley. I would also like to thank Professor Pieter Abbeel for getting me into machine learning. I would especially like to thank my mentor, Justin Fu for the guidance and support in all my research endeavours. Last, but not least, thank you to everyone on the 4th floor of Sutardja Dai Hall for the support and humor that decreased my research productivity as well as the tea from 7th floor Sutardja Dai Hall which has provided me with the caffeine that restored it.

And as always, to my family.

Chapter 1

Introduction

Reinforcement learning is proving to be a promising framework for solving many challenging problems, such as the strategy game Go and assisting in robot control tasks, but is often hindered by the difficulty to specify a reward function for agents to complete tasks. This creates a significant usability challenge, as poorly defined reward functions can produce unintended behaviors and writing reward functions for complex observations like images is unintuitive. On top of those challenges, the reward function engineered must be readily optimizable. One of the classical goals of artificial intelligence is to have agents respond to and complete tasks described by human-language. It is most natural for humans to express goals in language; these language commands can be grounded as a reward function. The objective is to allow users to convey goals to intelligent agents using human language-conditioned rewards trained through inverse reinforcement learning.

We are interested in solving the problem of learning a reward function which is optimizable by a reinforcement learning agent and is semantically grounded in structured language utterances. We propose Goal-Induced Inverse Reinforcement Learning (GIIRL), which uses a sample-based adversarial inverse reinforcement learning procedure to ground English language as reward functions. This allows for solving continuous action problems where a reward function may be difficult to specify. This algorithm provides the first step towards generalizing to tasks that can be learned from the same reward function, such as a robot putting away dishes or navigating an unknown area.

To translate language commands into reward functions, we propose to learn a reward that is a function of a language command and the agent's observation. This allows a natural representation of the reward function, one that is dependent on the command and state. In order to learn a feature representation for the language command we combine the feature-representation of pretrained utterance embeddings along with the agent's observations. From this augmented observation space, we propose to learn a reward using sample-based inverse reinforcement learning techniques. By using both the structured-language embeddings and the observation, this work is able to leverage expert demonstrations of a tasks to learn an underlying reward function. We show that a reward function can be learned to reproduce both the expert demonstrations and to generalize to different tasks within the environment.

In this work, we examine the method on a simple tabular environment to show that it can learn the true reward as well as on Internet browsing tasks to show that this method is capable of learning to navigate a complicated environment commanded by structured English commands. We also demonstrate that the reward function learned is optimizing for the correct behavior in each setting in the Internet-based tasks. Learning rewards from language commands is an important step towards improving human interfaces to intelligent agents, so that people may accurately convey their intentions and goals. Because language offers a more natural and intuitive means of communication for many people, non-domain experts will be able to interact and design reward functions for reinforcement agents. This work aims to provide a solution to this problem.

Chapter 2

Background and Related Works

2.1 Background

Reinforcement learning (RL) aims to solve problems in which an agent learns to maximize returns by interacting with its environment. The reinforcement learning environment can be formulated as a Markov Decision Process with states $\mathcal{S}$, actions $\mathcal{A}$, transitions T, rewards r, and discount factor γ. The agent navigates the environment that has dynamics $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ which takes an agent from a state $s \in \mathcal{S}$ by action $a \in \mathcal{A}$ to a new state s' with probability $T(s, a, s')$. From the environment, the agent obtains rewards $r(s, a, s')$ where $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$. In RL, the problem then becomes solving for a policy $\pi(a|s)$ that, given a state, outputs the probability distribution over the next action the agent should take to maximize its cumulative reward.

In recent years, deep reinforcement learning has removed the need to hand engineer features for policies, and thus RL has become a powerful framework for automating problems such as robot control tasks [Lev+15]. However, reward functions still often need to be hand engineered for good practical performance. Furthermore, deep learning procedures are highly sensitive to reward sparsity and magnitude, and thus engineering a good reward functions is particularly difficult, as a poorly defined reward function can produce unintended behaviors [Amo+16]. This work explores using natural language to communicate goal-conditioned rewards, which can be learned via solving the inverse reinforcement learning (IRL) problem [NR00; Rus98]. IRL refers to the problem of inferring an expert's reward function from demonstrations, which bypasses the need to hand engineer a reward function.

In this work, we hope to be able to ground a sparse reward, a goal-specification g, which is given as a natural language utterance (e.g., *"Go to the red square"*), to a reward function $r_g(s, a, s')$ from demonstrations of an expert completing instances of the goal g via the Inverse Reinforcement Learning problem framework. Because goals are inherently sparse, learning a reward function from demonstrations is preferred over having to optimize over a sparse reward function or hand engineering a dense reward function.

This problem is challenging in many ways: IRL is an inherently ill-defined problem, with many optimal policies and reward functions that can explain the same set of demonstrations [NR00]. Furthermore, using human utterances makes the problem more complex then checking for a goal state. Learning a reward function ultimately amounts to grounding a structured language goal into a general function which can be optimized by an RL agent.

2.2 Solving Language-Aware Tasks with Reinforcement Learning

There has been previous work on solving the problem of visual question answering, and language semantic featurization would be useful in context of learning a policy for goal-conditioned RL problems [And+17]. However, they do not attempt to solve a control task, and mostly rely on supervised learning.

Other previous works that accomplish linguistic direction following using reinforcement learning framework have used policy based approaches, but these tend to not generalize well due to the fact that policies would need to rely on zero-shot generalization on new tasks. Work by Mei, et. al. uses recurrent networks to focus on salient sentence regions when selecting an action [MBW15]. Another work by Misra et. al. grounds natural language to manipulation instructions by learning an energy function that outputs sequences of actions to complete the task [Kum+15].

A common approach for natural language following interfaces with RL is language conditioned policies that map a policy $\pi(a|s, g)$ from a state and language goal to an action. Work by Liu, Guu, Pasupat et. al. proposes using deep RL to solve web interface tasks with a goal given as a human utterance [Liu+18]. Their work proposes learning a policy $\pi(a|s, g)$ that conditions the policy on both the state and the goal utterance, with exploration constrained using a high-level "workflow" on allowable actions at each time step. A similar approach was taken in the work by Jia, et. al. where an architecture for RL-based web navigation was designed to solve goal conditioned tasks [JKB19]. This method learns separate networks for different action categories, which utilizes the goal condition language to parameterize the different Q-function modules.

2.3 Inverse Reinforcement Learning

Another approach to solving language-aware tasks is to learn a navigation policy from demonstrations, or apprenticeship learning. One example of which is Inverse Reinforcement learning, which aims to recover the reward from the demonstrations as well as imitate the expert behavior. Many approaches of IRL learn both a optimizable reward as well as a policy that induces the demonstration [NR00; AN04]. A recent framework is the GAIL algorithm, which uses adverserial context to learn a policy from the demonstrations using a discriminator and a policy [HE16]. However, this in itself does not learn a reoptimizable reward.

Other works using the adverserial framework show that an effective reward function can be learned from demonstrations. Previous work showed that by viewing the learned policy as a generator and the reward as a discriminator, a reward function that is disentangled from the actions can be learned [FLL17; QY19]. There has also been previous attempts at using an IRL framework for learning language-following tasks [Wil+18; Mac+15]. These works mostly relied on a formal reward specification language.

Perhaps the work most similar to ours is work by Fu et. al., as it solves for a reward grounded in language that can successfully be reoptimized [Fu+19]. But because it uses tabular MaxEnt inverse reinforcement learning, the model is limited to discrete state and action spaces with known dynamics. Additionally, it has limitations including the language structure must be in a certain specific semantic. Another work that uses goal conditioned reward learning is by Bahdanau, et. al. where a reward function is learned from the language goal using adversarial training [Bah+18]. This work demonstrates the usefulness of a generative framework for learning a more robust reward function on goal conditioned tasks, and the learned reward is able to generalize across different goals. However, because the reward function requires the train the discriminator to classify if a state is a goal state, by definition it offers only sparse rewards and the learned reward is difficult to optimize.

Chapter 3

Goal-Induced Inverse Reward Learning

3.1 Learning Rewards from Structured Languages

In this work, we introduce GIIRL ("Goal-Induced Inverse Reward learning"), which is a framework for learning a reward function for the Reinforcement Learning problem with goal specifications. Specifically, GIIRL solves the instruction-following task by optimizing a reward function concurrent with a policy. We formulate the problem as an Generative Adversarial problem [Goo+14], with the *generator network* being a goal-conditioned policy, $\pi_\theta(\cdot|s, g)$.

The policy is learned from interactions with the environment by adjusting parameters θ to maximize the maximum expected reward of a demonstration τ, $\mathbb{E}_{\pi_\theta}[\sum_{t\in\tau}\gamma^{t-1}\hat{r}_t+\alpha H(\pi_\theta)]$. Note that this is the objective of the maximum entropy Reinforcement Learning problem [Haa+17; Haa+18]. However, the stepwise reward $\hat{r}_t$ is not obtained from the environment, but obtained from the *discriminator network* which is a learned a reward model, D_ϕ. The model attempts to define a meaningful reward for the policy to train π_θ.

We formulate the *discriminator model* as solving the problem of classifying positive examples of expert task demonstration (s_E, a_E) given goal-instruction g_i from policy experiences (s_π, a_π) given goal-instruction g_π. Expert state-action pairs (s_E, a_E) paired with goal-instructions g_i are sampled from a fixed dataset $\mathcal{D}_E$; policy state-action pairs (s_π, a_π) are obtained from the policy π_θ interacting with the environment, paired with the instruction given to the policy.

Specifically, we formulate our *generator network* objective as learning a policy π_θ that maximizes expected return, $R_\pi(\theta)$ such that:

$$R_\pi(\theta) = \mathbb{E}_{\pi_\theta}[\sum_{t\in\tau}\gamma^{t-1}\hat{r}_t + \alpha H(\pi_\theta)], \tag{3.1}$$

and we formulate our *discriminator network* objective as learning a reward model D_ϕ that minimizes the loss, $L_D(\phi)$ such that:

$$L_D(\phi) = \mathbb{E}_{\pi_\theta}[-\log(1 - D_\phi(s_\pi, a_\pi|g_\pi))] + \mathbb{E}_{\mathcal{D}_E}[-\log D_\phi(s_E, a_E|g_i)] \tag{3.2}$$

The reward used is equal to the discriminator output, i.e. $\hat{r}_t = D_\phi(s_t, a_t|g_t)$. Thus, we can view D_ϕ as modeling the probability a given state-action pair is a positive label given the goal-instruction.

3.2 Goal-Induced Inverse Reinforcement Learning

A challenge of the Goal-Induced Inverse Reinforcement learning setting is that the reward function learned must be able to generalize across different goals within the same environment. This differs from the reward function learned in standard IRL problems, which are generally trained and evaluated on the same task and environment. Specifically, the GI-IRL reward function must return *different* rewards given different goal-instructions, where a state-action pair can have different rewards depending on what the instruction is at the moment.

We adopt the notation, task, denoted ξ, to be an instance of an environment $\mathcal{E}$ with the same dynamics and state space, but with a reward function that may differ between tasks. Each task is associated with a goal-instruction g, which is an English-language goal specification describing the task. In order to optimize the discriminator and the policy objectives, we sample tasks from the environment and optimize the reward function and policy for the task at each timestep. The algorithm is briefly summarized in Algorithm 1.

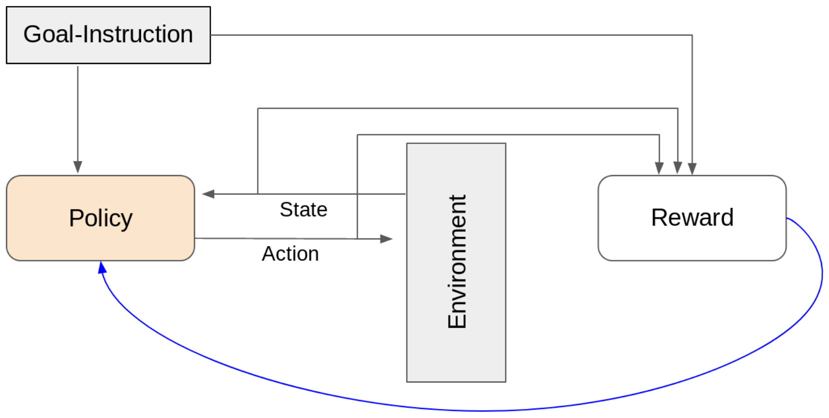
Algorithm 1: Goal induced inverse reinforcement learning (GIIRL)

```
Input: Environment $\mathcal{E}$
Populate a dataset of expert demonstrations and goal-instructions $\mathcal{D}_E$;
Initialize policy $\pi_\theta$ and discriminator $D_\phi$;
for step $i$ in $\{1, \dots, N\}$ do
    Sample task $\xi_i$ from $\mathcal{E}$ and corresponding goal-instruction $g_{i\pi}$;
    Collect rollout $\tau_{i\pi} = (s_{0\pi}, a_{0\pi} \dots s_{T\pi}, a_{T\pi})$ by executing $\pi_\theta$ on task $\xi_i$;
    Sample expert experiences $\tau_{iE}$ and goal-instructions $g_{iE}$ from $\mathcal{D}_E$;
    Train $D_\phi$ via binary logistic regression to classify $(\tau_{iE}, g_{iE})$ from $(\tau_{i\pi}, g_{i\pi})$;
    Update $r(s, a|g) \leftarrow \log D_\phi(s, a|g) - \log(1 - D_\phi(s, a|g))$;
    Train $\pi_\theta$ with respect to $r$ using any policy optimization method;
end
```

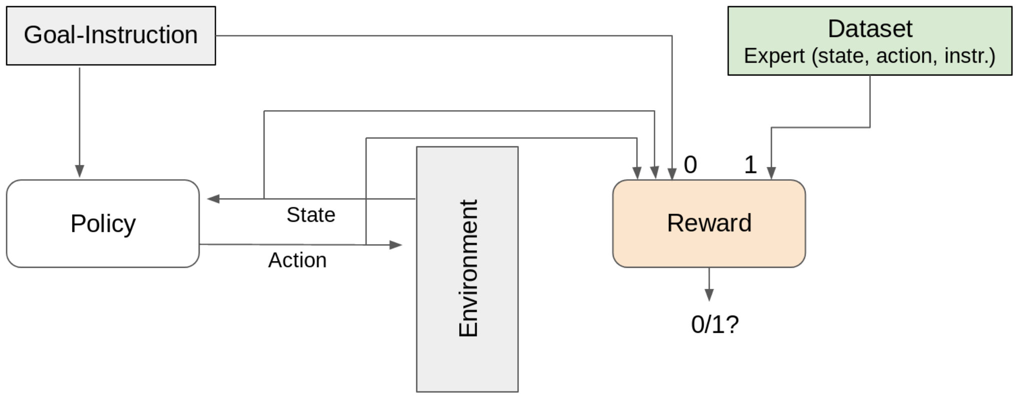
On a high level, we learn a language-aware reward function using the adversarial IRL framework adopted for multi-task setting and rewards that are goal-language conditioned models.

3.3 Architecture

Our algorithm architecture is shown below in Figure 3.1a and Figure 3.1b. The reward model and the policy are trained as separate units alternately. The policy model learns though interactions with the environment and trying to solve the task corresponding to the goal-instruction. The reward model is trained as a discriminator to distinguish the policy experiences from expert demonstrations.



(a) Training the policy network.



(b) Training the reward network.

Figure 3.1: GIIRL flow chart for training the reward and policy networks. The policy is trained from the goal-instruction and reward from the reward model (Figure (a)). The reward network is trained by learning a discriminator between the policy experience (state, action, instruction) and the expert (state, action, instruction) from a dataset (Figure (b)).

3.4 Language Embeddings

In this work, we incorporated various elements of Natural Language Processing in order to learn a reward function that is language-aware. For simple tasks with limited vocabulary, we selected to use one-hot encoding by using Bag of Words features. This had the benefits of ease of implementation.

However, for more complex language structures, we used the GloVe embeddings [PSM14]. GloVe embeddings has the feature that the cosine similarity between two word vectors provides an effective method for measuring the linguistic or semantic similarity of the corresponding words. This allowed for easy featurization of the goal-instruction utterances, which can then be used to train the reward model.

Chapter 4

Experiments

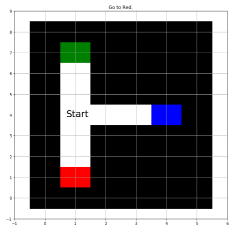
We evaluated our method within a tabular setting and a more complex, World-of-Bits environment. In our experiments we aim to answer the questions:

1. Can GIIRL learn rewards that are generalizable to different goal-instructions?
2. Is GIIRL able to induce a policy that imitates the optimal behavior?

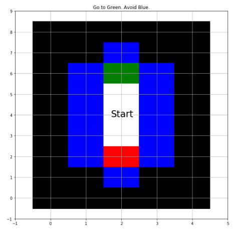
To answer 1, we evaluate the method on environments with different goal-instructions and show that GIIRL learns a reward function that changes depending on the language specification. To answer 2, we compared GIIRL to an imitation learning algorithm behavioral cloning as benchmark tasks. We also show that GIIRL learns a shaped reward from the underlying sparse true-rewards.

4.1 Tabular Setting

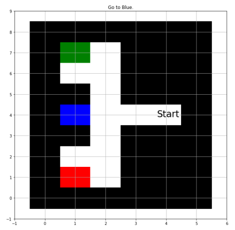
We begin our evaluations in a tabular setting on the Gridcraft environment shown in Figure 4.1. Each environment consists of different tasks, which is encoded as a English sentence. Each task has its own true reward which represents the sparse goal-specification from the task description.



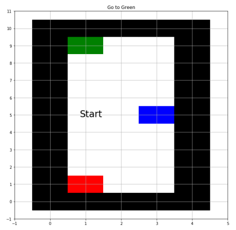
(a) “Go to Red”



(b) “Go to Green. Avoid Blue.”



(c) “Go to Blue”



(d) “Go to Green”

Figure 4.1: Different environments in the LanguageGridCraft and their corresponding task goal-instruction.

Reward Analysis

For this environment, we modeled the reward function as a deep neural network. The reward model takes in the goal-instruction features, the observation, and the agent action. The model, shown in Figure 4.2, takes as input the concatenation of the language featurization and the observation, and the one-hot representation of the action. It passes the language-observation unit through a multi-layer perceptron (MLP), and concatenates with the action vector which is itself passed through a MLP. The final result is passed through a last (MLP) which outputs the discriminator value.

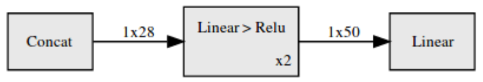
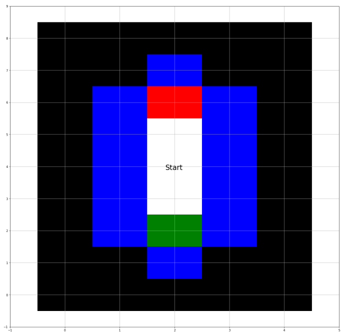
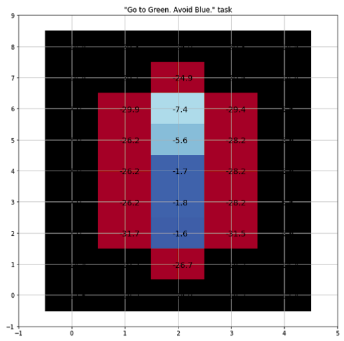


Figure 4.2: Tabular reward model.

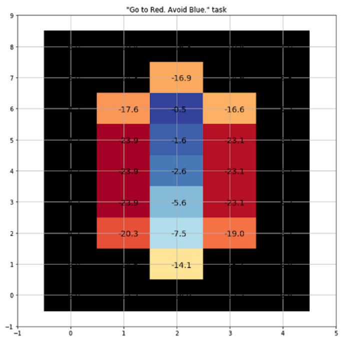
After using GIIRL to learn the reward model on the tabular setting, we visualized the average rewards obtained in each state for a LanguageGridCraft environment (Figure 4.7). We showed that the rewards learned maps to the true, underlying sparse reward function with the difference that it is shaped. Note from the figure, the rewards are higher closer to the goal, with a gradient outwards. This is desirable, as a shaped reward is usually easier for an RL agent to learn then a sparse reward. Furthermore, we showed that by changing the goal-instruction, the reward function outputs different values for the same state. The difference can be observed in Figure 4.3c and Figure 4.3b.



(a) True environment. It consists of Red, blue, and green regions.



(b) Heatmap of the reward function learned on the sentence "Go to Green. Avoid Blue".

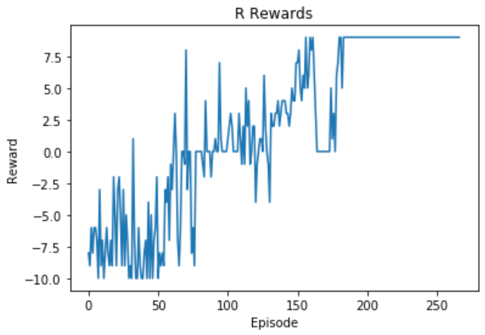


(c) Heatmap of the reward function learned on the sentence "Go to Red. Avoid Blue".

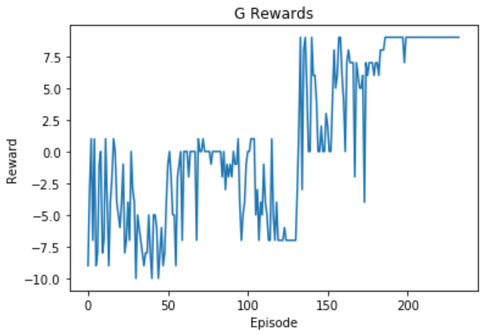
Figure 4.3: Figures of the learned reward function with different goal-instructions. Notice that the reward function learned is a more dense reward that solves the sparse goal specification. Darker blue represents a higher reward, and darker red represents a lower reward.

Policy Evaluation

We used the Soft-Q Learning algorithm as our policy optimization module [Haa+17]. We evaluated their performance when learning from the reward model learned using GIIRL. We adapted the algorithm for our multi-task setting by incorporating the featurized goal-instruction as part of the agent's observation. The results are shown below in Figure 4.4. We showed that the policy learned using the GIIRL framework is able to achieve optimal true reward for all tasks when the corresponding goal-instruction is given to the policy.



(a) True rewards obtained on task "Go to Red. Avoid Blue".



(b) True rewards obtained on task "Go to Green, Avoid Blue".

Figure 4.4: Plots of the true reward optimized from a Soft Q-Learning policy using the learned reward model. The policy was able to reach the optimal true reward.

4.2 Mini-World of Bits Environment

For further analysis of the method, we used the Mini-World of Bits Environment, which emulates solving a web-based task given a language goal-specification [Shi+17]. Each task contains a 160px-by-210px environment and a goal specified in text. The tasks return a single sparse reward at the end of the episode: either +1 (success) or -1 (failure). The agent has access to the environment via a Selenium web driver interface. Some of the environments that we worked with are shown in Figure 4.5.

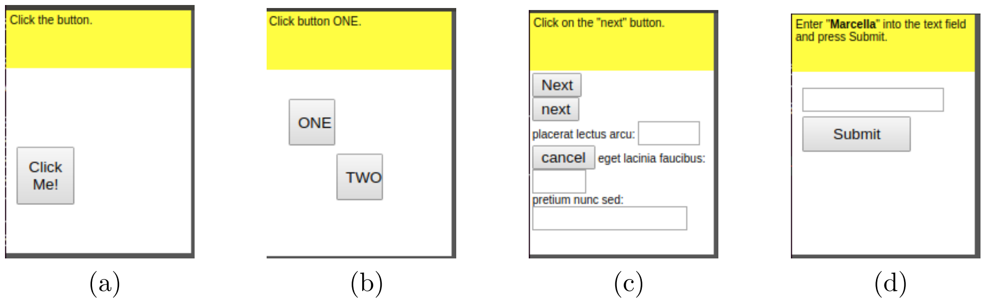


Figure 4.5: Some of the tasks in the Mini-World of Bits Environment; these are the four tasks this method was evaluated on.

This environment was significantly more challenging because the state space of a web-interface involves a mix of structured (e.g. HTML) and unstructured (e.g. natural language and images) inputs. Because the state space is very large and complicated, the algorithm will need to be flexible enough handle an infinite size state-space and learn a reasonable reward for solving such tasks. We designed a model and showed that GIIRL is able to navigate this complex environment.

Web Navigation Reward Model

The model that we selected for learning the policy on the Mini-World of Bits environment was the Workflow Guided Exploration Policy (WGE) [Liu+18]. Work by Liu, Guu, Pasupat et. al. on WGE showed that it achieved state-of-the-art results on web interface language-aware tasks.

We created our reward model shown in Figure 4.6, which is heavily inspired by DOM-Net proposed in the WGE paper. The rewards model is able to capture both spatial and hierarchical structure of the DOM tree of the web-interface, as well as the goal-instruction passed into it. Key features include the goal and DOM embedders, which are able to capture the interactions between language and DOM elements, respectively.

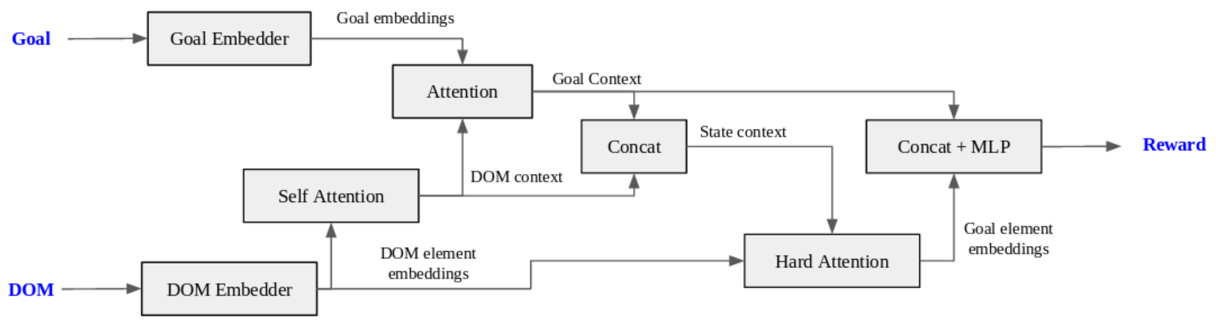


Figure 4.6: Mini World-of-Bits reward model.

Results

We evaluated the performance of GIIRL as an imitation learning algorithm on four tasks, *click-test*, *click-test-2*, *click-button*, and *enter-text*, which are each described in Table 4.1. The policy, when evaluated on the true rewards, obtain converges to the optimal policy. The results are shown in Figure 4.7.

Note that the GI-IRL algorithm is able to learn a reward which can solve for a two step task, which implies that it is able to learn more then a strict goal-specification from the language of the goal-instruction.

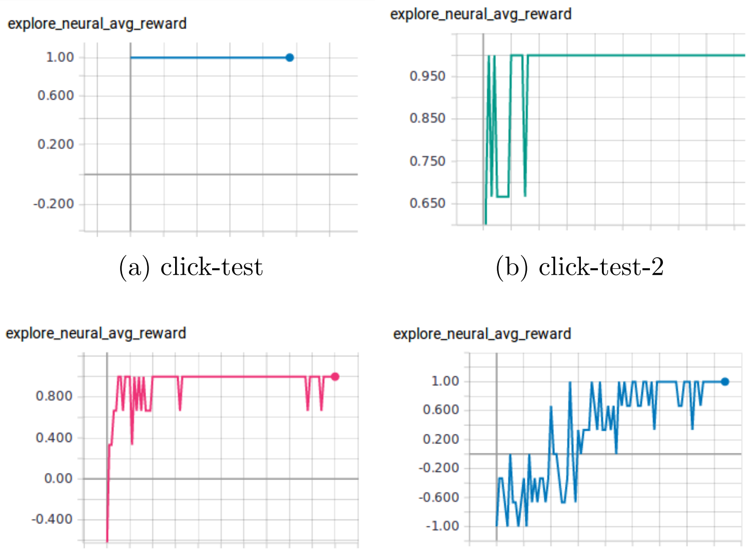


Figure 4.7: Plots of true reward the policy achieves by learning on the GIIRL reward verses algorithm step. GIIRL achieved optimal true reward though optimizing learned reward model on Mini-World of Bits Environment tasks.

Task	**Description**	**Steps**	**Success**
click-test	Click on the button	1	100
click-test-2	Click between two buttons	1	100
click-button	Click on a specific button	1	99
enter-text	Enter a specific text and click the submit button	2	94

Table 4.1: Description of each of the Mini-World of Bits tasks GIIRL was evaluated on, and the success rate the method achieved.

Comparison to Behavioral Cloning

As a baseline of comparison, we evaluated the results obtained by GIIRL to an imitation-learning algorithm Behavioral Cloning. As a metric for comparison, we used the percentages of successes (which occurs if the task obtains a score of +1) on the Mini World of Bits tasks. For all tasks, GIIRL was able to achieve perfect success rates on the test set. However, Behaviorial Cloning began to overfit on the examples pretty quickly, and would result in slightly lower success rates. As shown in Figure 4.8, for the more complex task *enter-text*, Behavioral cloning was only able to succeed on average 89% of the time, whereas GIIRL achieves 100% success rate.

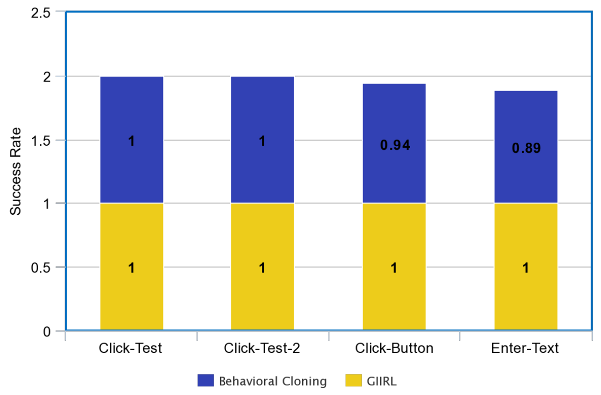


Figure 4.8: Comparison to Behavioral Cloning method on each of the tasks. These comparisons were done on averaging of a large test set.

Chapter 5

Conclusion and Future Work

5.1 Discussion

In this work, we present Goal-Induced Inverse Reinforcement Learning, an IRL framework that learns a transferable reward function and achieves good performance as compared to other imitation-learning algorithms. The GIIRL framework is able to learn rewards from a sparse, goal-specification language that is reshaped into a dense, optimizable reward. By learning the rewards in the IRL framework, we are able to obtain a more generalizable algorithm that is able to solve different tasks by changing just the goal specification. Indeed, this work showed that the reward function learned changes to match the task at hand, and can be toggled depending on the given goal-instruction. This shows that it learns a semantic mapping to the true, underlying reward function that the language intends. Furthermore, by training the policy and reward models jointly, we are able to efficiently obtain a policy that can perform on par with other imitation-learning policies. In this work, we showed that GIIRL shows comparable, if not better, results to Behavioral-Cloning algorithm, and is able to perform well under evaluations of the true reward.

A further appeal to the GIIRL algorithm is the fact that the reward model and the policy model are decoupled. This allows training of one to occur independently of the other, if need be. Thus, if the features of the task environment changes, finetuning the policy while optimizing on a frozen reward model can allow the policy to learn even for a more generalized setting. Furthermore, this work demonstrates the viability in using IRL algorithms as GIIRL is able to learn quickly, and when evaluated against the ground truth rewards, it is shown that the policy converges to the optimal policy. The speed at which the policy learns can be attributed to the fact that though general goal-specification task only contains sparse rewards, GIIRL learns a shaped reward which eases the optimization of the policy.

Overall, GIIRL proves that it is possible to learn generalizable reward functions from complex task environments. This provides many new opportunities for training language-aware agents as language annotated demonstrations are much more viable then hand engineering and implementing reward functions. This would enable a larger audience of people to use the powerful tools of Reinforcement Learning.

5.2 Future Directions of Work

The tasks that IRL problems can solve still remains mostly in simulated environments, and for language-instructed tasks specifically, there currently exists a large gap between what was presented in this paper and what may occur in "real world" environments. However, GIIRL proves to be a promising first step in this direction. Further work should focus on the effects of changing the reward architecture and analyzing how using the shaped reward compares in performance to optimizing the true, sparse reward. In addition, more experiments that slows the training of the reward model as the policy performance improves should be taken into consideration. A possible other direction could be to incorporate a human-in-the-loop aspect, where the policy is continually learning from human feedback. The experiments in this work used a static set of expert demonstrations, which can be improved by incorporating continual feedback. The results suggest that future work in the IRL framework will be able to incorporate this element.

Bibliography

[Amo+16] Dario Amodei et al. "Concrete Problems in AI Safety". In: *CoRR* abs/1606.06565 (2016). arXiv: `1606.06565`. URL: `http://arxiv.org/abs/1606.06565`.

[AN04] Pieter Abbeel and Andrew Y Ng. "Apprenticeship learning via inverse reinforcement learning". In: *Proceedings of the twenty-first international conference on Machine learning*. ACM. 2004, p. 1.

[And+17] Peter Anderson et al. "Bottom-Up and Top-Down Attention for Image Captioning and VQA". In: *CoRR* abs/1707.07998 (2017). arXiv: `1707.07998`. URL: `http://arxiv.org/abs/1707.07998`.

[Bah+18] Dzmitry Bahdanau et al. "Learning to Follow Language Instructions with Adversarial Reward Induction". In: *CoRR* abs/1806.01946 (2018). arXiv: `1806.01946`. URL: `http://arxiv.org/abs/1806.01946`.

[FLL17] Justin Fu, Katie Luo, and Sergey Levine. "Learning Robust Rewards with Adversarial Inverse Reinforcement Learning". In: *CoRR* abs/1710.11248 (2017). arXiv: `1710.11248`. URL: `http://arxiv.org/abs/1710.11248`.

[Fu+19] Justin Fu et al. "From Language to Goals: Inverse Reinforcement Learning for Vision-Based Instruction Following". In: *CoRR* abs/1902.07742 (2019). arXiv: `1902.07742`. URL: `http://arxiv.org/abs/1902.07742`.

[Goo+14] Ian Goodfellow et al. "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems 27*. Ed. by Z. Ghahramani et al. Curran Associates, Inc., 2014, pp. 2672–2680. URL: `http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf`.

[Haa+17] Tuomas Haarnoja et al. "Reinforcement Learning with Deep Energy-Based Policies". In: *CoRR* abs/1702.08165 (2017). arXiv: `1702.08165`. URL: `http://arxiv.org/abs/1702.08165`.

[Haa+18] Tuomas Haarnoja et al. "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor". In: *CoRR* abs/1801.01290 (2018). arXiv: `1801.01290`. URL: `http://arxiv.org/abs/1801.01290`.

[HE16] Jonathan Ho and Stefano Ermon. "Generative Adversarial Imitation Learning". In: *CoRR* abs/1606.03476 (2016). arXiv: `1606.03476`. URL: `http://arxiv.org/abs/1606.03476`.

[JKB19] Sheng Jia, Jamie Kiros, and Jimmy Ba. "DOM-Q-NET: Grounded RL on Structured Language". In: *CoRR* abs/1902.07257 (2019). arXiv: 1902.07257. URL: http://arxiv.org/abs/1902.07257.

[Kum+15] Dipendra Kumar Misra et al. "Tell Me Dave: Context-Sensitive Grounding of Natural Language to Manipulation Instructions". In: *The International Journal of Robotics Research* 35 (Nov. 2015). DOI: 10.1177/0278364915602060.

[Lev+15] Sergey Levine et al. "End-to-End Training of Deep Visuomotor Policies". In: *CoRR* abs/1504.00702 (2015). arXiv: 1504.00702. URL: http://arxiv.org/abs/1504.00702.

[Liu+18] Evan Zheran Liu et al. "Reinforcement Learning on Web Interfaces Using Workflow-Guided Exploration". In: *CoRR* abs/1802.08802 (2018). arXiv: 1802.08802. URL: http://arxiv.org/abs/1802.08802.

[Mac+15] James MacGlashan et al. "Grounding English Commands to Reward Functions". In: *Robotics: Science and Systems*. 2015.

[MBW15] Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. "Listen, Attend, and Walk: Neural Mapping of Navigational Instructions to Action Sequences". In: *CoRR* abs/1506.04089 (2015). arXiv: 1506.04089. URL: http://arxiv.org/abs/1506.04089.

[NR00] Andrew Y. Ng and Stuart J. Russell. "Algorithms for Inverse Reinforcement Learning". In: *Proceedings of the Seventeenth International Conference on Machine Learning*. ICML '00. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000, pp. 663–670. ISBN: 1-55860-707-2. URL: http://dl.acm.org/citation.cfm?id=645529.657801.

[PSM14] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. "GloVe: Global Vectors for Word Representation". In: *Empirical Methods in Natural Language Processing (EMNLP)*. 2014, pp. 1532–1543. URL: http://www.aclweb.org/anthology/D14-1162.

[QY19] Ahmed H. Qureshi and Michael C. Yip. "Adversarial Imitation via Variational Inverse Reinforcement Learning". In: *CoRR* abs/1809.06404 (2019).

[Rus98] Stuart Russell. "Learning Agents for Uncertain Environments (Extended Abstract)". In: *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*. COLT' 98. Madison, Wisconsin, USA: ACM, 1998, pp. 101–103. ISBN: 1-58113-057-0. DOI: 10.1145/279943.279964. URL: http://doi.acm.org/10.1145/279943.279964.

[Shi+17] Tianlin Shi et al. "World of Bits: An Open-Domain Platform for Web-Based Agents". In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. International Convention Centre, Sydney, Australia: PMLR, June 2017, pp. 3135–3144. URL: `http://proceedings.mlr.press/v70/shi17a.html`.

[Wil+18] Edward C. Williams et al. "Learning to Parse Natural Language to Grounded Reward Functions with Weak Supervision". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)* (2018), pp. 1–7.