Inverse Reinforcement Learning Tool for MiniGrid environment

Giulio Bazzanti Niccolò Biondi

Human Computer Interaction, 2020 March 5

Table of Contents

Possible Scenario

Persona

- Paolo Dibali
- Age 27
- Video Game Designer and non AI expert
- Paolo wants to create his video game

- Non-Player Characters (NPCs) approaches:
  - Handcrafted rules
  - Reinforcement Learning (RL)

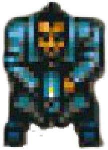
Video games and AI

- RL is used in video games to create NPCs

- Two problems:
  - NPCs with super-human abilities
  - NPCs with predictable behavior

Video games and AI

- RL is used in video games to create NPCs
- Two problems:
  - NPCs with super-human abilities
  - NPCs with predictable behavior

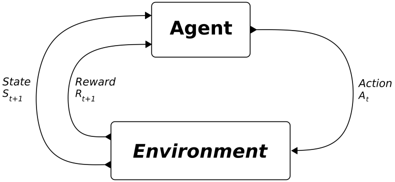


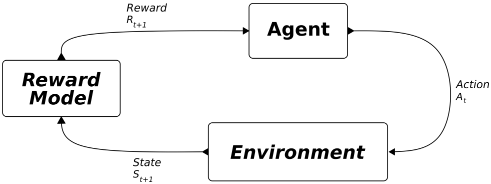
- **Solution:** Paolo can control the agent behavior $\Rightarrow$ IRL

Reinforcement Learning vs Inverse Reinforcement Learning

- RL agent takes environments rewards
- The agent learns the behavior through *trial-and-error* mechanism

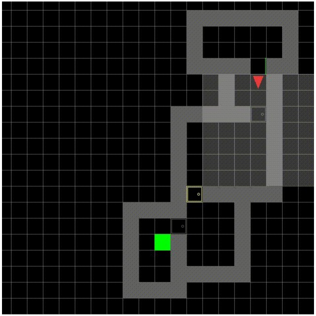
- IRL agent receives rewards from a Reward Model
- The user controls the agent behavior with preference feedback





IRL Benefits

- The user does not specify the Reward Function
- No sparse rewards
- Faster convergence to goal
- Human interaction



Proposed Method

Proposed method

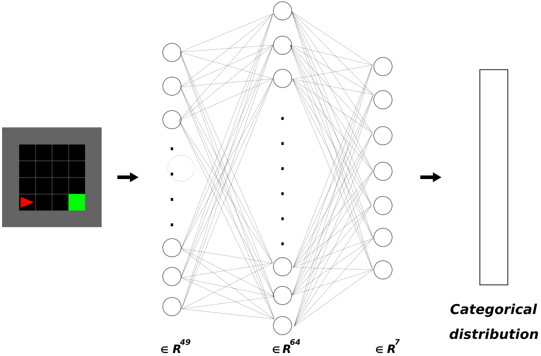
Algorithm 1 Training Protocol [1]

1: Run the policy in the environment and store "initial trajectories".
2: The annotator annotated all the "initial clips" and create the annotation buffer.
3: Pretrain the reward model from the annotation buffer.
4: **for** M epochs **do**
5: Train the policy in the environment for N episodes with rewards from the reward model.
6: Sample pairs of clips from the resulting trajectories.
7: The annotator labels the selected pairs and puts them in the annotation buffer.
8: Train the reward model for K batches from the annotation buffer.
9: **end for**

[1] Reward learning from human preferences and demonstrations in Atari, *Ibarz et all.*

Policy

- The policy receives the current agent state as input and probabilities of actions as output
- The next action is sampled from a Categorical distribution



Annotator

- The annotator gives preference feedback about pairs of clips
  - (0,1) (1,0) preferred clips
  - (0.5,0.5) indifferent labels
  - (0,0) discarded clips

Annotator

- The annotator gives preference feedback about pairs of clips
  - (0,1) (1,0) preferred clips
  - (0.5,0.5) indifferent labels
  - (0,0) discarded clips

- Human annotator vs Artificial annotator (*Oracle*)

Annotator

- The annotator gives preference feedback about pairs of clips
  - (0,1) (1,0) preferred clips
  - (0.5,0.5) indifferent labels
  - (0,0) discarded clips
- Human annotator vs Artificial annotator (*Oracle*)

- All the preferences are stored in the Annotation Buffer

Reward Model

- The Reward Model has to emulate the annotator labels
- It is trained to minimize the cross-entropy loss between predictions and labels

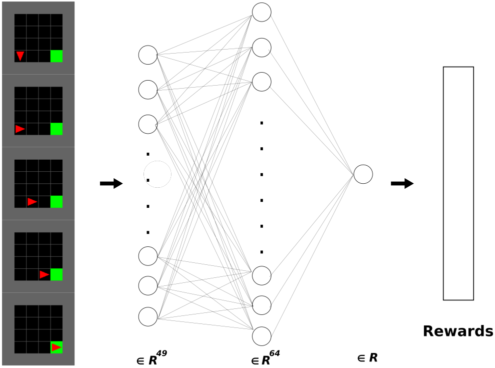
$$loss(\hat{r}) = -\sum_{(\sigma^1,\sigma^2,\mu)\in A} \mu(1) log(\hat{P}[\sigma^1 \succ \sigma^2]) + \mu(2) log(\hat{P}[\sigma^2 \succ \sigma^1])$$

- where

$$\hat{P}[\sigma^1 \succ \sigma^2] = \frac{exp \sum_{o\in\sigma^1} \hat{r}(o)}{exp \sum_{o\in\sigma^1} \hat{r}(o) + exp \sum_{o\in\sigma^2} \hat{r}(o)}$$

Reward Model

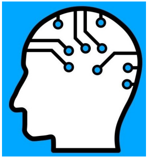
- The Reward Model predicts a list of rewards from a clip

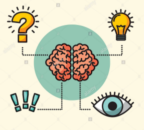


IRL Tool

Motivations

- The user is not a RL (or IRL) expert

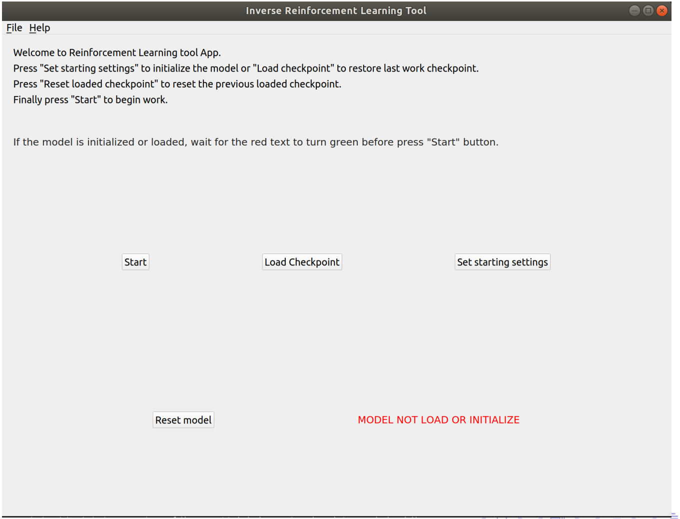




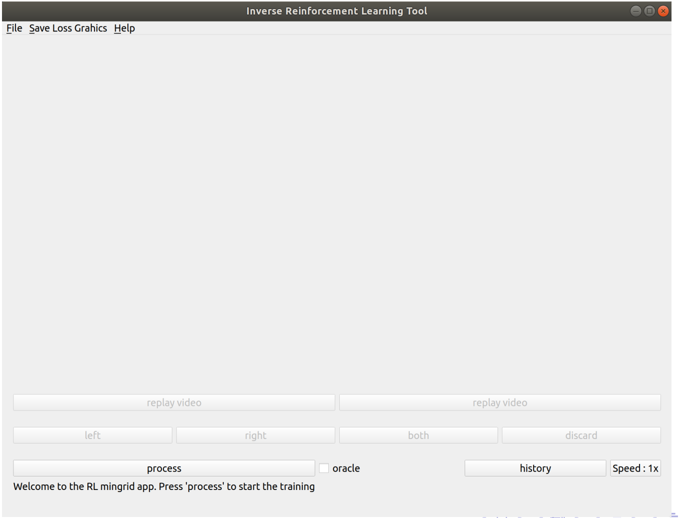
- Simplify the agent behavior understanding

- Better user experience

Set Up Window



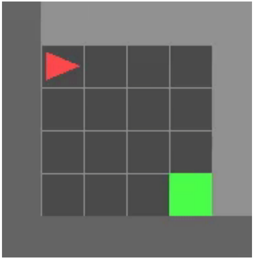
Training iteration



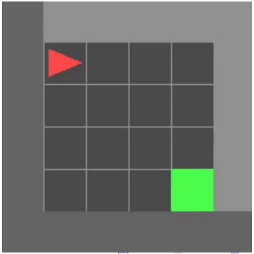
Experimental Results

MiniGrid Environment

- 6x6 Empty MiniGrid environment for experiments
- "Optimal" (maximize the discounted reward) trajectory achieved with RL



- Sub-optimal trajectory forced with IRL
- The user controls the agent behaviour



Tuning the components

- Policy
  - the agent has to reach easily the environment goal $\Rightarrow$ episode length 150
  - no negative goal reward or big values range $\Rightarrow$ standard deviation 0.5

Tuning the components

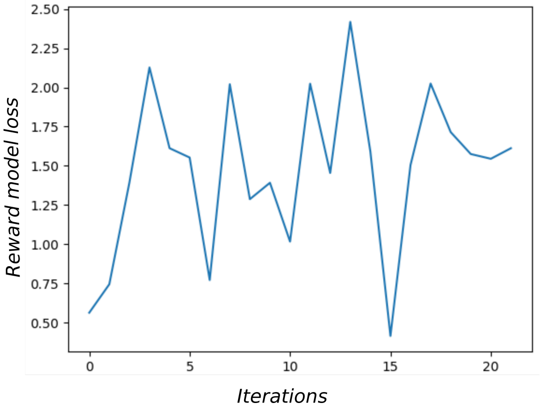
- Policy
  - the agent has to reach easily the environment goal $\Rightarrow$ episode length 150
  - no negative goal reward or big values range $\Rightarrow$ standard deviation 0.5
- Annotator
  - sparse vs dense Oracle

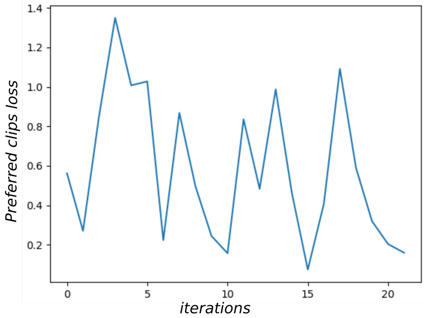
Tuning the components

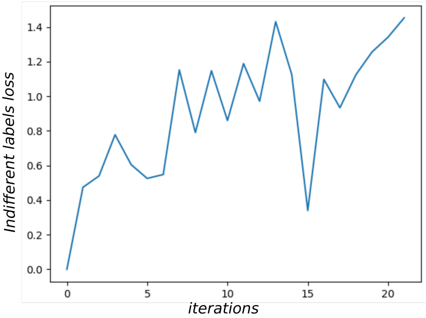
- Policy
  - the agent has to reach easily the environment goal $\Rightarrow$ episode length 150
  - no negative goal reward or big values range $\Rightarrow$ standard deviation 0.5
- Annotator
  - sparse vs dense Oracle
- Reward Model
  - variable K batch vs constant K batch
  - early stopping and decreasing annotations

Reward Model Loss

- The reward model loss grows during the training protocol.







- Small contribution to total loss
- Same range during all the training, decreasing trend

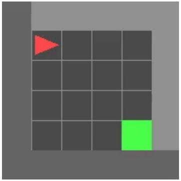
- Big contribution to total loss
- The number of those preferences makes the loss grow

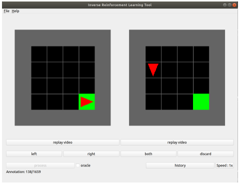
$$loss(\hat{r}) = -\sum_{(\sigma^1,\sigma^2,\mu)\in A} \mu(1)log(\hat{P}[\sigma^1 \succ \sigma^2]) + \mu(2)log(\hat{P}[\sigma^2 \succ \sigma^1])$$

Conclusions

Conclusions and Future Works

- For now ...
  - The agent learns sub-optimal trajectory
  - A not-expert user can build IRL system
  - Reward model emulates the human behavior





- ...then
  - More complex environment to increase IRL benefits
  - Process environment images with agent states (with CNN and/or LSTM)

Thanks for the attention