Inverse Reinforcement Learning to give desired behaviors to autonomous agents in MiniGrid

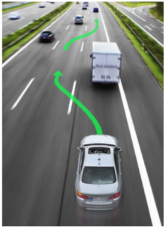
Human Computer Interaction
Federico Magnolfi & Francesca Del Lungo
Prof. Andrew D. Bagdanov

15 April 2020

Overview

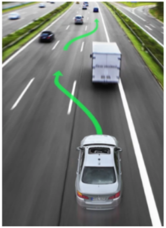
Scenarios

Real world



Scenarios

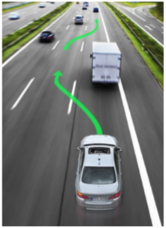
Real world



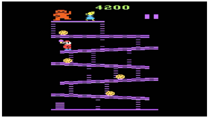
→ agent
→ environment
→ goal

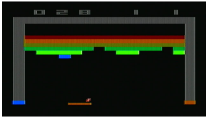
Scenarios

Real world



Computer games



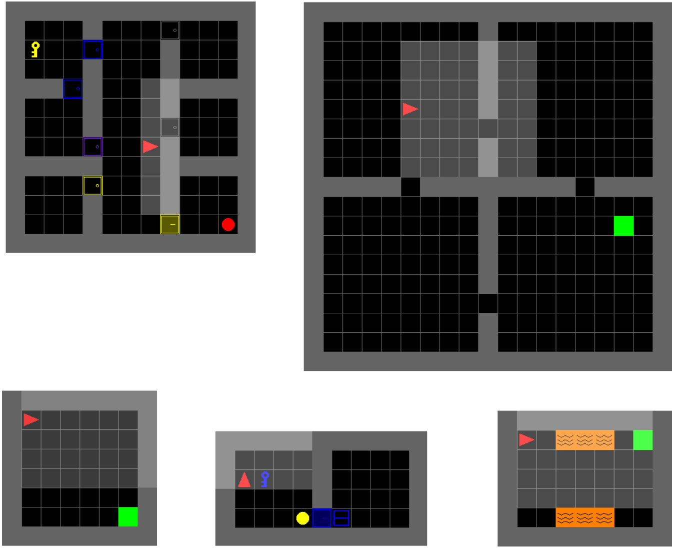


$\rightarrow$ agent

$\rightarrow$ environment

$\rightarrow$ goal

Considered environment: MiniGrid



Objectives

Objectives of this study:

- create agents that show **desired behaviors**...
- ...from **user demonstrations**
- create a **graphical tool** that can be used by non-experts

Objectives

Objectives of this study:

- create agents that show **desired behaviors**...
- ...from **user demonstrations**
- create a **graphical tool** that can be used by non-experts

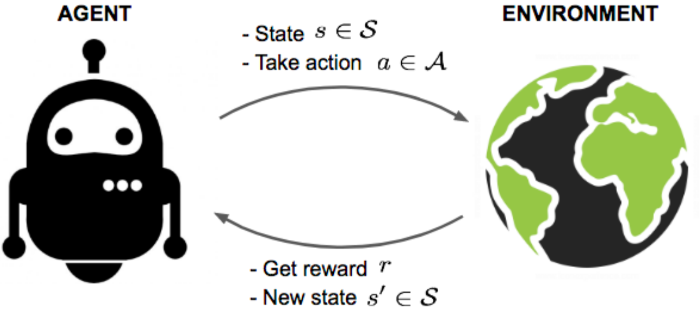
How

Using Inverse Reinforcement Learning, in particular the T-REX* approach

*: *Extrapolating Beyond Suboptimal Demonstrations via Inverse Reinforcement Learning from Observations* (Brown et al, 2019)

Reinforcement Learning
&
Inverse Reinforcement Learning

Reinforcement Learning



Reinforcement Learning

RL difficulties:

- design a reward that **induces** desired behaviours
- even more difficult for **non** Machine Learning **experts**

Reinforcement Learning

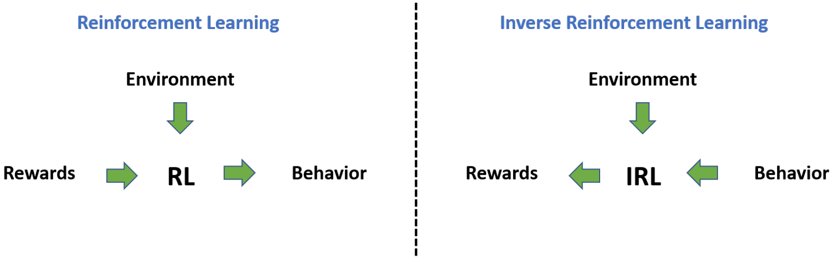
RL difficulties:

- design a reward that **induces** desired behaviours
- even more difficult for **non** Machine Learning **experts**

Possible learning-related problems:

- many **unsuccessful iterations**
- user **cannot change** agent behavior
- **long time** to train

Inverse Reinforcement Learning



Inverse Reinforcement Learning

Definitions

- $s \in S$: state
- $\tau \in S^n$: demonstration
- $R(s)$: reward given in the state s

We want to approximate the reward function $R(s)$ with a neural network, using the T-REX loss.

T-REX loss

Given a sequence of m demonstrations ranked from worst to best $\tau_1, ..., \tau_m$. When $j > i$, we want:

$$\sum_{s \in \tau_j} R_\theta(s) > \sum_{s \in \tau_i} R_\theta(s)$$

T-REX loss

Given a sequence of m demonstrations ranked from worst to best $\tau_1, ..., \tau_m$. When $j > i$, we want:

$$R_\theta(\tau_j) = \sum_{s \in \tau_j} R_\theta(s) > \sum_{s \in \tau_i} R_\theta(s) = R_\theta(\tau_i)$$

T-REX loss

Given a sequence of m demonstrations ranked from worst to best $\tau_1, ..., \tau_m$. When $j > i$, we want:

$$R_\theta(\tau_j) = \sum_{s \in \tau_j} R_\theta(s) > \sum_{s \in \tau_i} R_\theta(s) = R_\theta(\tau_i)$$

Probability of trajectory j being better than trajectory i:

$$p(\tau_j, \tau_i) = \frac{\exp R_\theta(\tau_j)}{\exp R_\theta(\tau_i) + \exp R_\theta(\tau_j)}$$

T-REX loss

Given a sequence of m demonstrations ranked from worst to best $\tau_1, ..., \tau_m$. When $j > i$, we want:

$$R_\theta(\tau_j) = \sum_{s \in \tau_j} R_\theta(s) > \sum_{s \in \tau_i} R_\theta(s) = R_\theta(\tau_i)$$

Probability of trajectory j being better than trajectory i:

$$p(\tau_j, \tau_i) = \frac{\exp R_\theta(\tau_j)}{\exp R_\theta(\tau_i) + \exp R_\theta(\tau_j)}$$

T-REX loss is a *cross-entropy* over pairs:

$$\mathcal{L}(\theta) = -\sum_{j>i} \log p(\tau_j, \tau_i)$$

Putting all together

$\{\text{demonstrations}\} \rightarrow$ Reward $R_\theta(s)$

Putting all together

$\{\text{demonstrations}\} \rightarrow$ Reward $R_\theta(s) \rightarrow$ Discounted Reward $D_R(s)$

Discounted Reward $D_R(s)$

$$D_R(s_t) = \sum_{k \leq t} R_\theta(s_k) * \gamma^{t-k}$$

Putting all together

{demonstrations}$\rightarrow$ Reward $R_\theta(s)$ $\rightarrow$ Discounted Reward $D_R(s)$ $\rightarrow$ Policy $\pi(s)$

Discounted Reward $D_R(s)$

$$D_R(s_t) = \sum_{k \leq t} R_\theta(s_k) * \gamma^{t-k}$$

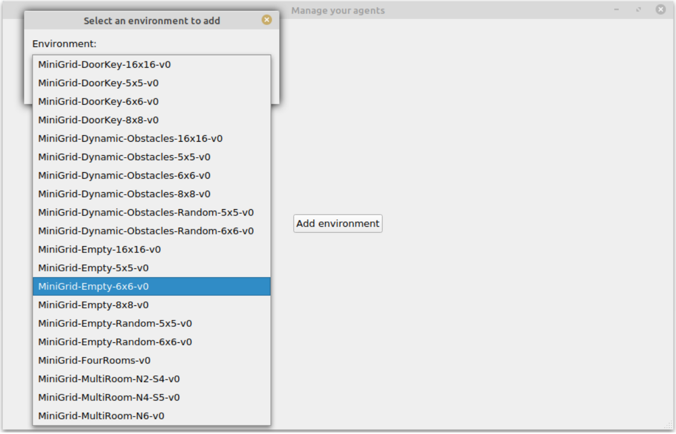
Policy $\pi(s)$

- $\pi(s)$: distribution over actions, determined by state s
- loss: $-log(p_a) * D_R(s)$

Graphical Application

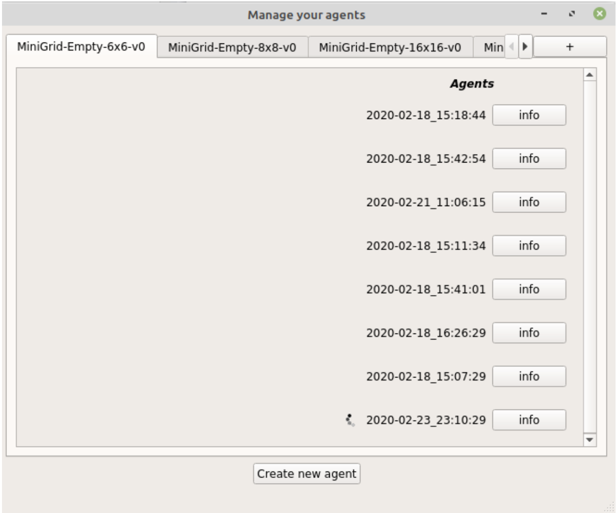
Initial window

- choose one **environment**
- **20** different environments available



Agents management

- one **tab** for each environment
- list of **existing agents** of the selected environment
- button for **new agent** creation
- ⁘ agent training in **progress**

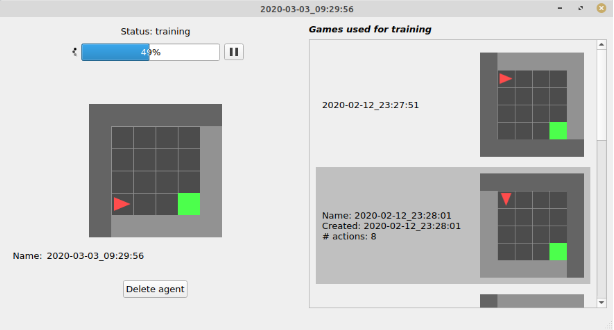


New agent

- **All games** list: games played so far
- **Ranked games** list: selected games to train the agent
- **Arrows** to move games between the lists and within the ranking list

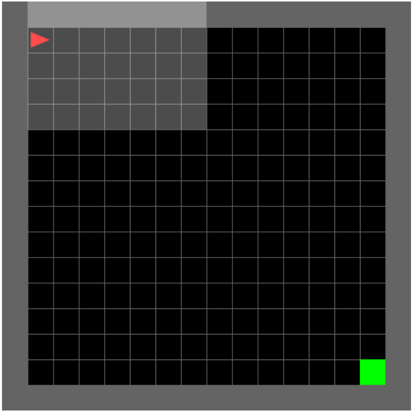
Agent details

- **Training status**
- **Play-pause training**: on pause visualize previous behaviours
- Agent **playing**
- Used **demonstrations**



Experiments

MiniGrid environment



Agent state

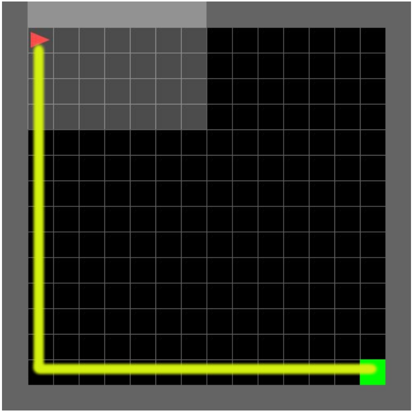
3x7x7 integer tensor

- objects
- colors
- objects states

Environment

Empty 16x16

Desired trajectory



Objective

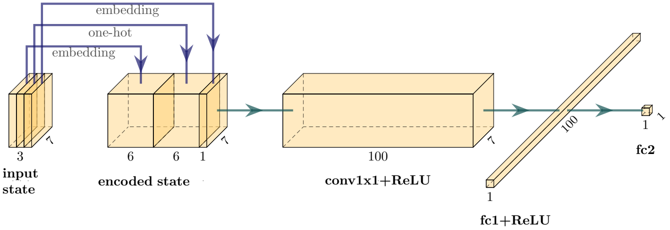
Force a sub-optimal trajectory

Trajectories

- 11, human created
- ranked from best to worst

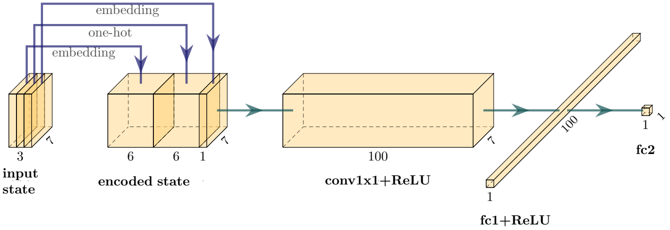
Networks

reward net

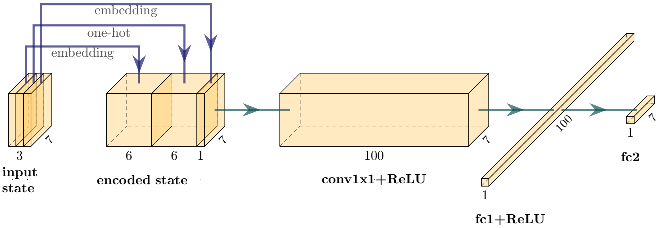


Networks

reward net



policy net



Experiments

Evaluation criteria

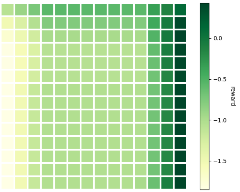
- **ordering_quality**: % of correctly ranked pairs of trajectories*
- **heatmap**: shows the rewards obtained in each cell of the grid
- **policy's behaviour** after training

*: a pair (τ_j, τ_i) with $j > i$ is correctly ranked *iff*:

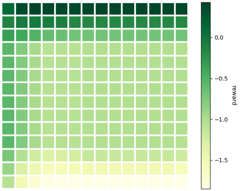
$$R_\theta(\tau_j) > R_\theta(\tau_i)$$

Results

Heatmaps

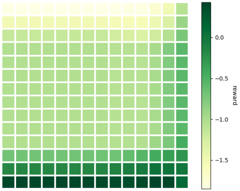


UP

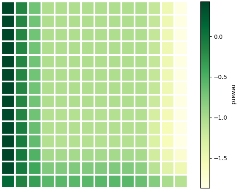


LEFT

RIGHT



DOWN



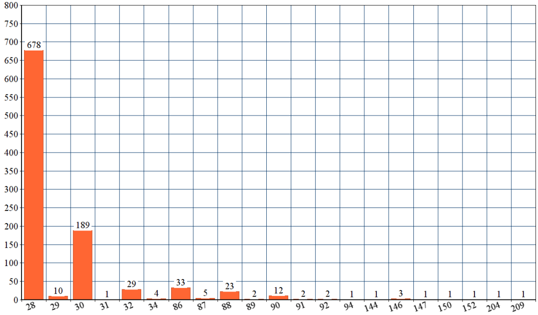
Directions with highest reward

For the cells in the desired path:
highest reward for desired direction

↓	←	←	←	←	←	←	←	←	←	←	←	←	←
↓	←	←	←	←	←	←	←	←	←	←	←	↑	↑
↓	↓	←	←	←	←	←	←	←	←	←	↑	↑	↑
↓	↓	↓	→	→	→	→	→	↓	↓	↓	↑	↑	↑
↓	↓	↓	→	→	→	→	→	↓	↓	↓	↑	↑	↑
↓	↓	↓	→	→	→	→	→	↓	↓	↓	↑	↑	↑
↓	↓	↓	→	→	→	→	→	↓	↓	↓	↑	↑	↑
↓	↓	↓	→	→	→	→	→	↓	↓	↓	↑	↑	↑
↓	↓	↓	→	→	→	→	→	←	←	←	↑	↑	↑
↓	↓	↓	→	→	→	→	→	←	←	←	↑	↑	↑
↓	↓	↓	→	→	→	→	←	←	←	←	↑	↑	↑
↓	↓	↓	→	→	→	→	→	→	→	→	→	↑	↑
↓	↓	→	→	→	→	→	→	→	→	→	→	→	↑
→	→	→	→	→	→	→	→	→	→	→	→	→	↑

Policy trained with learned reward

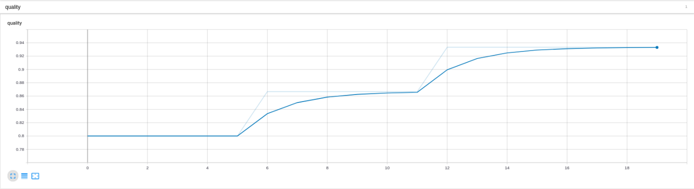
Histogram of lengths



- 1000 sampled trajectories
- desired trajectory: **67.8%**
- desired trajectory + spin: 18.9%
- sometimes more "laps"

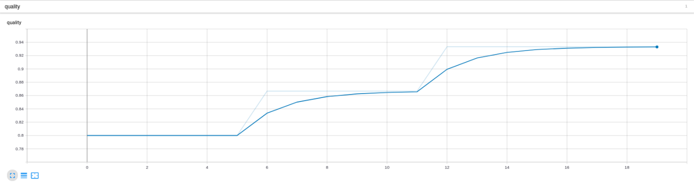
Trainings details

`reward net` *ordering quality* $\rightarrow$ 94%

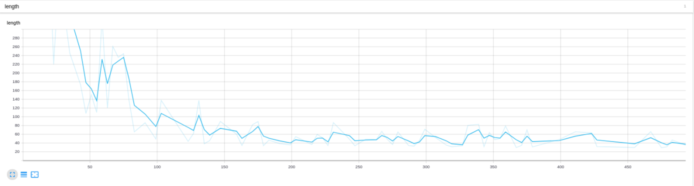


Trainings details

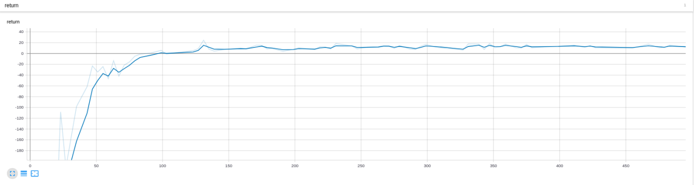
`reward net`
ordering quality
$\rightarrow$ 94%



`policy net`
trajectory length
$\rightarrow$ $\sim$30



`policy net`
trajectory return
$\rightarrow$ $>$0



Conclusions

Conclusions

Graphical tool

- allows to create **agents** from **demonstrations**
- **no** *Machine Learning* knowledge **required**

Experimental results

- **learning the reward** for simple trajectories is a **feasible** task
- learned rewards are **consistent** and **dense**

Future works

Graphical tool

- make the **graphical tool** more generic and customizable by the user
- visualize the **same agent** on **different environments**

Learning

- adjust the **policy**
- neural network architecture with **memory**

The end

Thanks for the attention