# Inverse Reinforcement Learning

## Notes

**Pay attention to this chapter. The opponent is expert in this field. Formulate carefully!**

## Body

In this section, we would like to introduce Inverse Reinforcement Learning (IRL) as an extension of Reinforcement Learning (RL) technique. In our opinion, IRL can make a useful addition to already existing approaches (more on this in section {link}) how to employ huge data sets with examples of professional play in games like StarCraft and bootstrap some of the expert level domain knowledge to AI agents. Due to the complexity of our domain and feasibility, we currently restrict ourselves to the simplest cases. Therefore we consider only IRL in Finite State Space using Markov Decision Processes (MDPs) presented in [1] and [2] for our experiments. First consideration can also be motivated by the way humans think when playing a game. For example in StarCraft domain there exist many guides (most well known is [3]) how to play in given situation (state).

### Reinforcement Learning and Markov Decision Process

Reinforcement Learning is very popular machine learning technique to solve situations where the agent does not know the payoff it will receive for its actions, so at first it has to explore the environment by taking random steps to learn what is an actual payoff for any action in given situation. To some extent, this can be seen as a very simple form of trial and error learning similar to learning experienced by humans during their lives.

In RL agent’s goal is to choose right action in any states he is currently in to maximize his feature discounted reward. That is, to find optimal **policy**. To give a formal definition of problem we need to formalize (finite) Markov decision process (MDP) first. An MDP notation in [2] is given as a tuple , where

* S is a finite set of **states** with cardinality N
* represents set of k **actions**
* are the state **transitions probabilities** upon taking action a in state s
* is the **discount factor**
* is the **reinforcment (reward) function** is bounded in absolute value by

Further on we will write notation R(s, a) as R(s) for simplicity in equations. A policy is defined as any mapping , and the **value function** for a policy is given as the expectation over the distribution of the sequence of states we visit by executing policy :



**Q-function** is given as:



where notation means the expectation with respect to state s’ distributed according to . Assuming that MDP can model all agent's decision one can find optimal policy such that is maximized. Algorithms finding optimal policies are based on two basic properties of MDPs. All for an MDP satisfy (Bellman Equations):





moreover, the policy is optimal policy for an MDP if and only if, for all holds (Bellman Optimality):



Illustration of RL process to obtain optimal policy is on figure {link}. Inputting environment model (MDP) and specifying reward function R(s) one can get optimal policy trough Reinforcement Learning.

{RL framework figure}

### Inverse Reinforcement Learning

Inverse Reinforcement Learning (IRL) is a form of Apprenticeship Learning in MDP where in contrast to RL reward function is not known. Instead, few trajectories demonstrating expert performing the task to learn are available. The goal of IRL is then to use those trajectories and learn reward function that can explain expert’s behavior. Recovering reward function first and using it to generate desirable behavior (trough RL) is what makes IRL standout from other supervised learning techniques as most of them is learning policy as a mapping from states to actions right away. According to [2]: “reward function provides a much more parsimonious description of behavior.” As is mention in [2]: “the entire field of RL is founded on the presupposition that reward function, rather than the policy, is the most succinct, robust, and transferable definition of the task.” IRL is especially useful when we are attempting to learn intelligent agent that can behave successfully in our domain where is hard to define reward function; it is extremely labor or designer has only limited knowledge of it. RTS games are excellent examples of that as results of actions are not immediately observable. In figure {link} one can see IRL framework in contrast to RL framework illustrated on figure {link} we are using optimal policy to learn reward function.

{IRL framework figure}

### Solving Inverse Reinforcement Learning problem in Finite State Space

As was shown in [2] a solution to IRL problem in Finite State Space (for finite MDP) can be found trough **linear programming** (LP). LP can be formulated based on the characterization of the **Solution Set** consisting of all reward functions for which given policy is optimal. Following theorem from [2] characterizes the Solution Set:

Let a finite state space S, a set of actions , transition probability matricies , and a discount factor be given. Then the policy π given by is optimal if and only if, for all , the reward R satisfies



Proof of {link} is based on substitution equation {link 1.4} to {link 1.5} from theorem written above and can be found in [2]. Before giving LP formulation, two problems with this characterization of the Solution Set need to be taken into account. First, any constant vector is always a solution of R, and second, many solutions satisfying equation {link 1.6} exist so choosing one R may impose challenge. One of the ways described in [2] to overcome the problem of choosing R is to demand that selected R makes given policy optimal and favors solutions that make any deviation from costly as possible. From all R functions satisfying {link 1.6} and , we pick one maximizing:



Then we can formulate the optimization problem as LP which we want to maximize:

