Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

b-it Bonn-Aachen International Center for Information Technology

Institute for AI and Autonomous Systems

Inverse Reinforcement Learning

An Overview

Dr. Alex Mitrevski
Master of Autonomous Systems

Structure

Artificial Intelligence Review (2022) 55:4307–4346
https://doi.org/10.1007/s10462-021-10108-x

A survey of inverse reinforcement learning
Stephen Adams[1] · Tyler Cody[1] · Peter A. Beling[1]

Learning agents for uncertain environments (extended abstract)

Stuart Russell*
Computer Science Division
University of California
Berkeley, CA 94720
russell@cs.berkeley.edu

- Motivation behind inverse reinforcement learning
- Reminder: Mathematical optimisation
- Inverse reinforcement learning methods

Motivation

Reward Functions in Reinforcement Learning

- Recall that, in our discussion on robot learning (for manipulation), we defined a task family of t tasks as a collection of Markov decision processes (MDPs):

$$P(\mathcal{M}) = \{(S_i, A, \mathcal{T}_i, R_i, C_i, \gamma) \mid 1 \leq i \leq \mathrm{t}\}$$

Reward Functions in Reinforcement Learning

- Recall that, in our discussion on robot learning (for manipulation), we defined a task family of t tasks as a collection of Markov decision processes (MDPs):

$$P(\mathcal{M}) = \{(S_i, A, \mathcal{T}_i, R_i, C_i, \gamma) \mid 1 \leq i \leq \mathrm{t}\}$$

- One MDP element that is essential for successful reinforcement learning is the **reward function**
  - The RL learning objective is that of **maximising the expected return** $E\left[\sum_t r_{i,t}\left(\boldsymbol{s}_t, \boldsymbol{a}_t\right)\right]$

Reward Functions in Reinforcement Learning

- Recall that, in our discussion on robot learning (for manipulation), we defined a task family of t tasks as a collection of Markov decision processes (MDPs):

$$P(\mathcal{M}) = \{(S_i, A, \mathcal{T}_i, R_i, C_i, \gamma) \mid 1 \leq i \leq \mathrm{t}\}$$

- One MDP element that is essential for successful reinforcement learning is the **reward function**
  - The RL learning objective is that of **maximising the expected return** $E\left[\sum_t r_{i,t}\left(\boldsymbol{s}_t, \boldsymbol{a}_t\right)\right]$
- But **defining a good reward function**, aka reward engineering, **is a challenging problem**
  - Learning with sparse rewards, on the other hand, is typically sample inefficient

Reward Functions in Reinforcement Learning

- Recall that, in our discussion on robot learning (for manipulation), we defined a task family of t tasks as a collection of Markov decision processes (MDPs):

$$P(\mathcal{M}) = \{(S_i, A, \mathcal{T}_i, R_i, C_i, \gamma) \mid 1 \leq i \leq \mathrm{t}\}$$

- One MDP element that is essential for successful reinforcement learning is the **reward function**
  - The RL learning objective is that of **maximising the expected return** $E\left[\sum_t r_{i,t}\left(\boldsymbol{s}_t, \boldsymbol{a}_t\right)\right]$
- But **defining a good reward function**, aka reward engineering, **is a challenging problem**
  - Learning with sparse rewards, on the other hand, is typically sample inefficient
- The challenges of reward engineering stem from multiple conceptual questions:

Reward Functions in Reinforcement Learning

- Recall that, in our discussion on robot learning (for manipulation), we defined a task family of t tasks as a collection of Markov decision processes (MDPs):

$$P(\mathcal{M}) = \{(S_i, A, \mathcal{T}_i, R_i, C_i, \gamma) \mid 1 \leq i \leq \mathrm{t}\}$$

- One MDP element that is essential for successful reinforcement learning is the **reward function**
  - The RL learning objective is that of **maximising the expected return** $E\left[\sum_t r_{i,t}\left(\boldsymbol{s}_t, \boldsymbol{a}_t\right)\right]$
- But **defining a good reward function**, aka reward engineering, **is a challenging problem**
  - Learning with sparse rewards, on the other hand, is typically sample inefficient
- The challenges of reward engineering stem from multiple conceptual questions:
  - **How should we come up with reward functions?**

Reward Functions in Reinforcement Learning

- Recall that, in our discussion on robot learning (for manipulation), we defined a task family of t tasks as a collection of Markov decision processes (MDPs):

$$P(\mathcal{M}) = \{(S_i, A, \mathcal{T}_i, R_i, C_i, \gamma) \mid 1 \leq i \leq \mathrm{t}\}$$

- One MDP element that is essential for successful reinforcement learning is the **reward function**
  - The RL learning objective is that of **maximising the expected return** $E\left[\sum_t r_{i,t}\left(\boldsymbol{s}_t, \boldsymbol{a}_t\right)\right]$
- But **defining a good reward function**, aka reward engineering, **is a challenging problem**
  - Learning with sparse rewards, on the other hand, is typically sample inefficient
- The challenges of reward engineering stem from multiple conceptual questions:
  - **How should we come up with reward functions?**
  - **What makes a good reward function?**

Reward Functions in Reinforcement Learning

- Recall that, in our discussion on robot learning (for manipulation), we defined a task family of t tasks as a collection of Markov decision processes (MDPs):

$$P(\mathcal{M}) = \{(S_i, A, \mathcal{T}_i, R_i, C_i, \gamma) \mid 1 \leq i \leq \mathrm{t}\}$$

- One MDP element that is essential for successful reinforcement learning is the **reward function**
  - The RL learning objective is that of **maximising the expected return** $E\left[\sum_t r_{i,t}\left(\boldsymbol{s}_t, \boldsymbol{a}_t\right)\right]$
- But **defining a good reward function**, aka reward engineering, **is a challenging problem**
  - Learning with sparse rewards, on the other hand, is typically sample inefficient
- The challenges of reward engineering stem from multiple conceptual questions:
  - **How should we come up with reward functions?**
  - **What makes a good reward function?**
  - **(Why) Do we need reward functions in the first place?**

Reward Function Composition Example

- Suppose that we want to learn a policy for grasping objects

Reward Function Composition Example

- Suppose that we want to learn a policy for grasping objects
- A reward function for this problem may be written as **a linear combination of the form**

$$R(\boldsymbol{s}_t, \boldsymbol{a}_t) = w_1 \mathbb{1}_{grasp} + w_2 \mathbb{1}_{drop} + w_3 \mathbb{1}_{collision} + w_4 \|\boldsymbol{v}\| + w_5$$

Reward Function Composition Example

- Suppose that we want to learn a policy for grasping objects
- A reward function for this problem may be written as **a linear combination of the form**

$$R(\boldsymbol{s}_t, \boldsymbol{a}_t) = \underbrace{w_1 \mathbb{1}_{grasp}}_{\text{A large positive term for grasping the object}} + \underbrace{w_2 \mathbb{1}_{drop}}_{\text{A large negative term for dropping the object}} + \underbrace{w_3 \mathbb{1}_{collision}}_{\text{A negative term for every collision with another object}} + \underbrace{w_4 \|\boldsymbol{v}\|}_{\text{A negative term that increases with increasing speed}} + \underbrace{w_5}_{\text{A negative term for every additional time step}}$$

Reward Function Composition Example

- Suppose that we want to learn a policy for grasping objects
- A reward function for this problem may be written as **a linear combination of the form**

$$R(\boldsymbol{s}_t, \boldsymbol{a}_t) = \underbrace{w_1 \mathbb{1}_{grasp}}_{\text{A large positive term for grasping the object}} + \underbrace{w_2 \mathbb{1}_{drop}}_{\text{A large negative term for dropping the object}} + \underbrace{w_3 \mathbb{1}_{collision}}_{\text{A negative term for every collision with another object}} + \underbrace{w_4 \|\boldsymbol{v}\|}_{\text{A negative term that increases with increasing speed}} + \underbrace{w_5}_{\text{A negative term for every additional time step}}$$

- As you may imagine, coming up with such a reward is **a rather involved and sometimes seemingly arbitrary process**
  - Setting the weights is particularly challenging — the learning process can be very sensitive to those

Learning from Experts

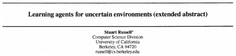
- A few lectures ago, we looked at learning from demonstrations, which enables a robot to learn given expert data
- The setting in which learning is done based on an expert is sometimes referred to as **apprenticeship learning**
- In our previous lecture, we only looked at learning from demonstration for learning execution policies, but **why not use demonstrations to learn reward functions**?
- The problem of extracting a reward function from expert demonstrations is called **inverse reinforcement learning**

Why Do We Need Rewards?


Artificial Intelligence Review (2022) 55:4307–4346
https://doi.org/10.1007/s10462-021-10108-x

A survey of inverse reinforcement learning

Stephen Adams[1] · Tyler Cody[1] · Peter A. Beling[1]


Learning agents for uncertain environments (extended abstract)

Stuart Russell*
Computer Science Division
University of California
Berkeley, CA 94720
russell@cs.berkeley.edu

- When learning from expert demonstrations, we can decide **not to bother with a reward function** and simply **learn the policy that a person is following**

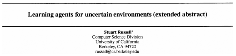
Why Do We Need Rewards?


Artificial Intelligence Review (2022) 55:4307–4346
https://doi.org/10.1007/s10462-021-10108-x

A survey of inverse reinforcement learning

Stephen Adams[1] · Tyler Cody[1] · Peter A. Beling[1]

Check for updates


Learning agents for uncertain environments (extended abstract)

Stuart Russell*
Computer Science Division
University of California
Berkeley, CA 94720
russell@cs.berkeley.edu

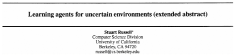
- When learning from expert demonstrations, we can decide **not to bother with a reward function** and simply **learn the policy that a person is following**
- This is reasonable if we want a robot to learn how to perform a particular problem, but **does not lead to a representation that is transferrable** to different — even related — problems
  - By acquiring a policy, we learn **what to do and how**, but not **why something is done**

Why Do We Need Rewards?


Artificial Intelligence Review (2022) 55:4307–4346
https://doi.org/10.1007/s10462-021-10108-x

A survey of inverse reinforcement learning

Stephen Adams[1] · Tyler Cody[1] · Peter A. Beling[1]


Learning agents for uncertain environments (extended abstract)

Stuart Russell*
Computer Science Division
University of California
Berkeley, CA 94720
russell@cs.berkeley.edu

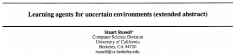
- When learning from expert demonstrations, we can decide **not to bother with a reward function** and simply **learn the policy that a person is following**
- This is reasonable if we want a robot to learn how to perform a particular problem, but **does not lead to a representation that is transferrable** to different — even related — problems
  - By acquiring a policy, we learn **what to do and how**, but not **why something is done**
- The behaviour of a robot that follows a given reward function is also (conceptually) more straightforward to understand than the behaviour of a robot that simply follows a given policy

Why Do We Need Rewards?


Artificial Intelligence Review (2022) 55:4307–4346
https://doi.org/10.1007/s10462-021-10108-x

A survey of inverse reinforcement learning

Stephen Adams[1] · Tyler Cody[1] · Peter A. Beling[1]


Learning agents for uncertain environments (extended abstract)

Stuart Russell*
Computer Science Division
University of California
Berkeley, CA 94720
russell@cs.berkeley.edu

- When learning from expert demonstrations, we can decide **not to bother with a reward function** and simply **learn the policy that a person is following**
- This is reasonable if we want a robot to learn how to perform a particular problem, but **does not lead to a representation that is transferrable** to different — even related — problems
  - By acquiring a policy, we learn **what to do and how**, but not **why something is done**
- The behaviour of a robot that follows a given reward function is also (conceptually) more straightforward to understand than the behaviour of a robot that simply follows a given policy

> "The reward function should usually be a simple monotonic function of the current sensory inputs, and thus may be much simpler than the direct decision mapping itself. That is, the most compact and hence robustly learnable representation of expert behavior may be the reward function." (Russell, 1998)

What Makes a Good Reward Function?

There are various properties that are desirable for reward functions

What Makes a Good Reward Function?

There are various properties that are desirable for reward functions

Consistency

A reward function should be representative of the knowledge that is implicitly provided by the demonstrations — the reward should be consistent with the expert demonstrations

What Makes a Good Reward Function?

There are various properties that are desirable for reward functions

Consistency

A reward function should be representative of the knowledge that is implicitly provided by the demonstrations — the reward should be consistent with the expert demonstrations

Simplicity

Occam's razor is a commonly applied principle in machine learning and applies to the problem of defining rewards as well — a reward should be as simple as possible

What Makes a Good Reward Function?

There are various properties that are desirable for reward functions

Consistency

A reward function should be representative of the knowledge that is implicitly provided by the demonstrations — the reward should be consistent with the expert demonstrations

Simplicity

Occam's razor is a commonly applied principle in machine learning and applies to the problem of defining rewards as well — a reward should be as simple as possible

Generalisability

A reward function should be defined so that it is not only applicable for a single problem setting, but so that it is robust to changes in the setting

Reminder: Mathematical Optimisation

Mathematical Optimisation

- **Unconstrained optimisation** problems are often relatively simple to solve (e.g. using Newton's method or gradient descent), but may not be sufficient to describe a problem of interest

- In the lecture on safe learning, we saw that we need to solve **constrained optimisation** problems; as we will see shortly, this is also the case in inverse reinforcement learning

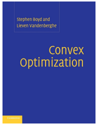
- A general constrained optimisation problem can be written as

$$
\begin{array}{ll}
\text{minimise} & f(\boldsymbol{x}) \\
\text{subject to} & g_i(\boldsymbol{x}) \leq 0,\ 1 \leq i \leq m \\
 & h_i(\boldsymbol{x}) = 0,\ 1 \leq i \leq p
\end{array}
$$

- Often, both the objective function f and the constraints are of a particular form (e.g. linear or convex), which can be solved using **linear and convex programming**, respectively

- The following few slides are there to remind you about mathematical optimisation so that you are not surprised by some of the notation later on

Linear Optimisation

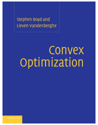


- Linear optimisation is a problem of **optimising a linear objective function that is subject to linear constraints**
- A linear optimisation problem has the following general form:

$$\begin{array}{ll} \text{minimise} & \boldsymbol{a}^T \boldsymbol{x} \\ \text{subject to} & \boldsymbol{c}_i^T \boldsymbol{x} \leq b_i,\ 1 \leq i \leq m \end{array}$$

- Linear optimisation problems can be solved using the **simplex method** or using **interior-point methods**

Convex Optimisation

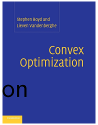


- The problem of **optimising a convex objective function that is subject to convex constraints** is called convex optimisation
  - For a convex function f, it holds that $f(\alpha x + \beta y) \leq \alpha f(x) + \beta f(y)$ for $\alpha, \beta \geq 0, \alpha + \beta = 1$
- A convex optimisation problem has the following general form:

$$\begin{array}{ll} \text{minimise} & f(\boldsymbol{x}) \\ \text{subject to} & g_i(\boldsymbol{x}) \leq b_i, \; 1 \leq i \leq m \end{array}$$

- Interior-point methods can also be used for solving convex optimisation problems

Lagrangian and the Dual Problem



- In optimisation, the method of **Lagrange multipliers** is often used to rewrite the optimisation problem and obtain a **dual representation** that can be solved more readily

- Let us particularly consider a general optimisation problem of the form

$$\begin{array}{ll} \text{minimise} & f(\boldsymbol{x}) \\ \text{subject to} & g_i(\boldsymbol{x}) \leq 0, \; 1 \leq i \leq m \\ & h_i(\boldsymbol{x}) = 0, \; 1 \leq i \leq p \end{array}$$

- The **Lagrangian** of this problem is defined as the function

$$L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) = f(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i g_i(\boldsymbol{x}) + \sum_{i=1}^{p} \nu_i h_i(\boldsymbol{x})$$

  where $\lambda_i, 1 \leq i \leq m$ and $\nu_i, 1 \leq i \leq p$ are called **Lagrange multipliers**

- A solution to the original problem can then be found by the following optimisation problem:

$$\begin{array}{ll} \text{maximise} & \inf_{\boldsymbol{x}} L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) \\ \text{subject to} & \boldsymbol{\lambda} \succeq 0 \end{array}$$

Inverse Reinforcement Learning Methods

IRL Objective

Learning agents for uncertain environments (extended abstract)

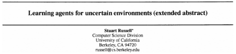
Stuart Russell*
Computer Science Division
University of California
Berkeley, CA 94720
russell@cs.berkeley.edu

- The objective of inverse reinforcement learning is simple to state in words: **a reward function that summarises the desired behaviour of a robot needs to be extracted**

"**Given**

- measurements of an agent's behaviour over time, in a variety of circumstances
- measurements of the sensory inputs to that agent
- a model of the physical environment (including the agent's body)

Determine the reward function that the agent is optimizing." (Russell, 1998)


Learning agents for uncertain environments (extended abstract)

Stuart Russell*
Computer Science Division
University of California
Berkeley, CA 94720
russell@cs.berkeley.edu

IRL Objective

- The objective of inverse reinforcement learning is simple to state in words: **a reward function that summarises the desired behaviour of a robot needs to be extracted**

"**Given**

- measurements of an agent's behaviour over time, in a variety of circumstances
- measurements of the sensory inputs to that agent
- a model of the physical environment (including the agent's body)

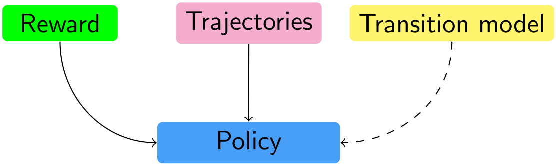
Determine the reward function that the agent is optimizing." (Russell, 1998)

- Typically, learning is done given **demonstrations** $\mathrm{T} = \{\mathrm{T}_1, ..., \mathrm{T}_M\}$, each of which has the form

$$\mathrm{T}_i = \left((\boldsymbol{s}_1, \boldsymbol{a}_1), ..., \left(\boldsymbol{s}_{\mathrm{T}_i^m}, \boldsymbol{a}_{\mathrm{T}_i^m}\right) \right)$$

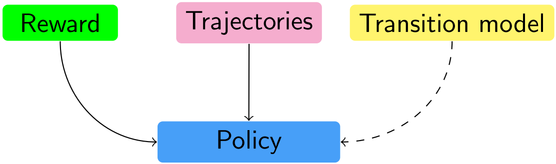
Reinforcement Learning vs. Inverse Reinforcement Learning

Reinforcement learning (RL)

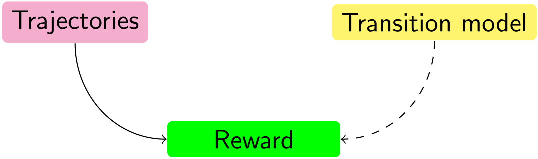


Reinforcement Learning vs. Inverse Reinforcement Learning

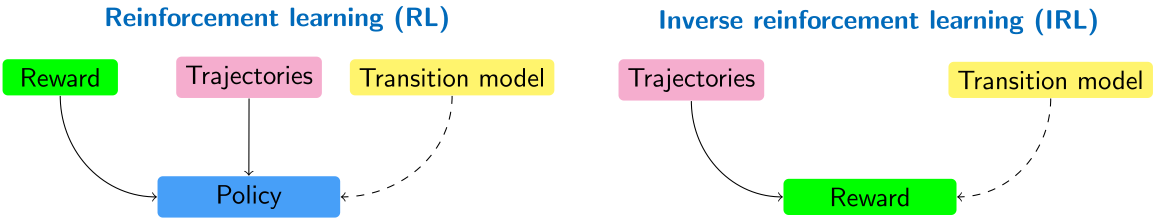
Reinforcement learning (RL)



Inverse reinforcement learning (IRL)



Reinforcement Learning vs. Inverse Reinforcement Learning



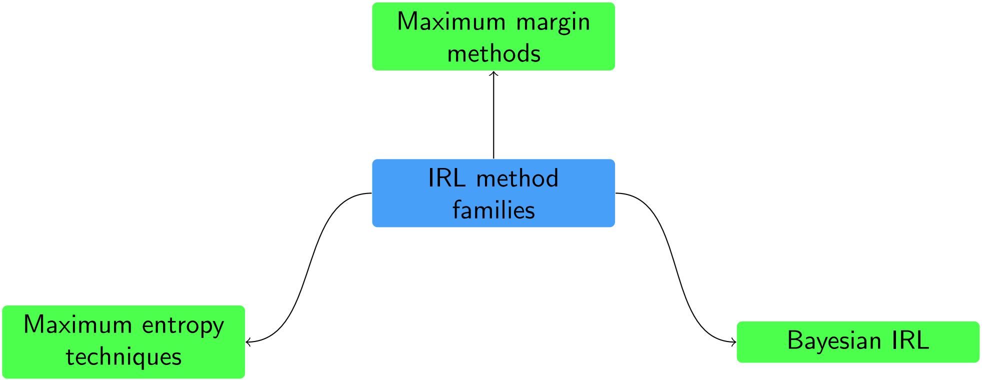
Many IRL methods are iterative, where RL is used in an internal loop that learns a policy given a reward hypothesis

IRL Methods Overview

Artificial Intelligence Review (2022) 55:4307–4346
https://doi.org/10.1007/s10462-021-10108-x

A survey of inverse reinforcement learning

Stephen Adams[1] · Tyler Cody[1] · Peter A. Beling[1]

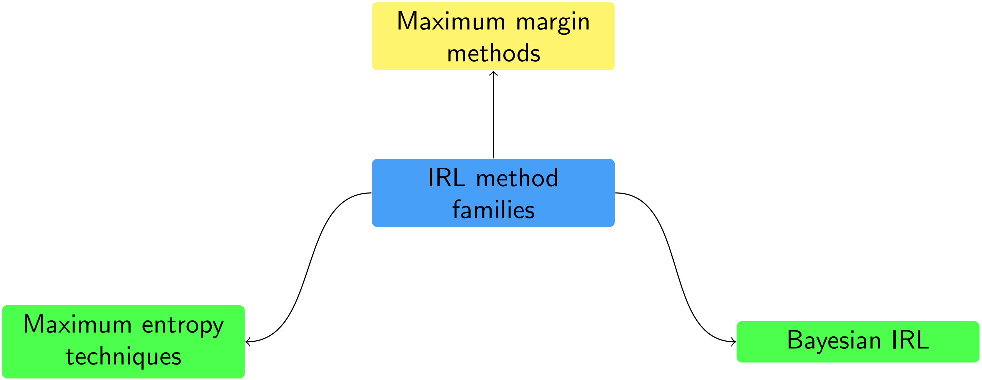


IRL Methods Overview

Artificial Intelligence Review (2022) 55:4307–4346
https://doi.org/10.1007/s10462-021-10108-x

A survey of inverse reinforcement learning

Stephen Adams[1] · Tyler Cody[1] · Peter A. Beling[1]



Maximum Margin Methods

Artificial Intelligence Review (2022) 55:4307–4346
https://doi.org/10.1007/s10462-021-10108-x

A survey of inverse reinforcement learning

Stephen Adams[1] · Tyler Cody[1] · Peter A. Beling[1]

- The objective of max margin methods is to define a reward function that maximises **a margin between an optimal policy and other policies**

Artificial Intelligence Review (2022) 55:4307–4346
https://doi.org/10.1007/s10462-021-10108-x
A survey of inverse reinforcement learning
Stephen Adams[1] · Tyler Cody[1] · Peter A. Beling[1]

Maximum Margin Methods

- The objective of max margin methods is to define a reward function that maximises **a margin between an optimal policy and other policies**
- Conceptually, the idea is similar to what is used in support vector machines

Maximum Margin Methods

Artificial Intelligence Review (2022) 55:4307–4346
https://doi.org/10.1007/s10462-021-10108-x
A survey of inverse reinforcement learning
Stephen Adams[1] · Tyler Cody[1] · Peter A. Beling[1]

- The objective of max margin methods is to define a reward function that maximises **a margin between an optimal policy and other policies**
- Conceptually, the idea is similar to what is used in support vector machines
- Max margin methods are typically **defined by a linear or a quadratic optimisation problem**

Maximum Margin Methods

Artificial Intelligence Review (2022) 55:4307–4346
https://doi.org/10.1007/s10462-021-10108-x
A survey of inverse reinforcement learning
Stephen Adams[1] · Tyler Cody[1] · Peter A. Beling[1]

- The objective of max margin methods is to define a reward function that maximises **a margin between an optimal policy and other policies**
- Conceptually, the idea is similar to what is used in support vector machines
- Max margin methods are typically **defined by a linear or a quadratic optimisation problem**
- The earliest max margin methods were proposed in
  - A. Y. Ng and S. J. Russell, "Algorithms for Inverse Reinforcement Learning," in *Proc. 17th Int. Conf. Machine Learning (ICML)*, 2000, pp. 663–670.
  - P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proc. 21st Int. Conf. Machine learning*, 2004.

  We will briefly look at some of these on the next few slides

Artificial Intelligence Review (2022) 55:4307–4346
https://doi.org/10.1007/s10462-021-10108-x
A survey of inverse reinforcement learning
Stephen Adams[1] · Tyler Cody[1] · Peter A. Beling[1]

Estimation with a Known Transition Model and Policy

Maximum Margin Methods

- Let us assume that we are given **a continuous state space S** and that the reward function R is **a linear combination of state features $\phi_k(\boldsymbol{s})$ with weights w_k**:

$$R(\boldsymbol{s}, \boldsymbol{a}) = \sum_{k=1}^{N} w_k \phi_k(\boldsymbol{s})$$

Artificial Intelligence Review (2022) 55:4307–4346
https://doi.org/10.1007/s10462-021-10108-x
A survey of inverse reinforcement learning
Stephen Adams[1] · Tyler Cody[1] · Peter A. Beling[1]

Estimation with a Known Transition Model and Policy

Maximum Margin Methods

- Let us assume that we are given **a continuous state space S** and that the reward function R is **a linear combination of state features $\phi_k(\boldsymbol{s})$ with weights w_k**:

$$R(\boldsymbol{s}, \boldsymbol{a}) = \sum_{k=1}^{N} w_k \phi_k(\boldsymbol{s})$$

- If this is the case, the value function can be written as

$$V^{\pi}(\boldsymbol{s}) = \sum_{k=1}^{N} w_k V_k^{\pi}(\boldsymbol{s})$$

Artificial Intelligence Review (2022) 55:4307–4346
https://doi.org/10.1007/s10462-021-10108-x
A survey of inverse reinforcement learning
Stephen Adams[1] · Tyler Cody[1] · Peter A. Beling[1]

Estimation with a Known Transition Model and Policy

Maximum Margin Methods

- Let us assume that we are given **a continuous state space S** and that the reward function R is **a linear combination of state features $\phi_k(\boldsymbol{s})$ with weights w_k**:

$$R(\boldsymbol{s}, \boldsymbol{a}) = \sum_{k=1}^{N} w_k \phi_k(\boldsymbol{s})$$

- If this is the case, the value function can be written as

$$V^\pi(\boldsymbol{s}) = \sum_{k=1}^{N} w_k V_k^\pi(\boldsymbol{s})$$

- In this case, $\pi = a_1$ **is optimal if its induced expected value for any action is at least as good as the value of any other action**:

$$\underset{\boldsymbol{s}' \sim P(\boldsymbol{s}' | \boldsymbol{s}, \boldsymbol{a}_1)}{E} \left[V^\pi(\boldsymbol{s}') \right] \geq \underset{\boldsymbol{s}' \sim P(\boldsymbol{s}' | \boldsymbol{s}, \boldsymbol{a})}{E} \left[V^\pi(\boldsymbol{s}') \right], \forall \boldsymbol{s} \in S, \forall \boldsymbol{a} \in A \setminus \boldsymbol{a}_1$$

Estimation with a Known Transition Model and Policy

Artificial Intelligence Review (2022) 55:4307–4346
https://doi.org/10.1007/s10462-021-10108-x
A survey of inverse reinforcement learning
Stephen Adams[1] · Tyler Cody[1] · Peter A. Beling[1]

Maximum Margin Methods

- Let us assume that we are given **a continuous state space S** and that the reward function R is **a linear combination of state features $\phi_k(\boldsymbol{s})$ with weights w_k**:

$$R(\boldsymbol{s}, \boldsymbol{a}) = \sum_{k=1}^{N} w_k \phi_k(\boldsymbol{s})$$

- If this is the case, the value function can be written as

$$V^{\pi}(\boldsymbol{s}) = \sum_{k=1}^{N} w_k V_k^{\pi}(\boldsymbol{s})$$

- In this case, $\pi = a_1$ **is optimal if its induced expected value for any action is at least as good as the value of any other action**:

$$\underset{\boldsymbol{s}' \sim P(\boldsymbol{s}' | \boldsymbol{s}, \boldsymbol{a}_1)}{E} \left[V^{\pi}(\boldsymbol{s}') \right] \geq \underset{\boldsymbol{s}' \sim P(\boldsymbol{s}' | \boldsymbol{s}, \boldsymbol{a})}{E} \left[V^{\pi}(\boldsymbol{s}') \right], \; \forall \, \boldsymbol{s} \in S, \; \forall \, \boldsymbol{a} \in A \setminus \boldsymbol{a}_1$$

- The optimisation problem here is given as

$$\begin{aligned} \max \sum_{\boldsymbol{s} \in S_0} \min_{\boldsymbol{a} \in A \setminus \boldsymbol{a}_1} \quad & p\left(\underset{\boldsymbol{s}' \sim P(\boldsymbol{s}' | \boldsymbol{s}, \boldsymbol{a}_1)}{E} \left[V^{\pi}(\boldsymbol{s}') \right] - \underset{\boldsymbol{s}' \sim P(\boldsymbol{s}' | \boldsymbol{s}, \boldsymbol{a})}{E} \left[V^{\pi}(\boldsymbol{s}') \right] \right) \\ \text{such that} \quad & |w_k| \leq 1, \; 1 \leq k \leq N \end{aligned}$$

where S_0 is a state subspace, and $p(x) = x$ if $x \geq 0$, otherwise $p(x) = 2x$

Reward Estimation Given Example Trajectories

Maximum Margin Methods

Artificial Intelligence Review (2022) 55:4307–4346
https://doi.org/10.1007/s10462-021-10108-x

A survey of inverse reinforcement learning

Stephen Adams[1] · Tyler Cody[1] · Peter A. Beling[1]

- In a more general case, we have demonstrations $\mathrm{T} = \{\mathrm{T}_1, ..., \mathrm{T}_M\}$ and the objective here is to **find a reward R that maximises the expected value $E\left[V^{\pi}(s_0)\right]$**, for s_0 belonging to an initial state distribution $\mathcal{D}$

Reward Estimation Given Example Trajectories

Artificial Intelligence Review (2022) 55:4307–4346
https://doi.org/10.1007/s10462-021-10108-x
A survey of inverse reinforcement learning
Stephen Adams[1] · Tyler Cody[1] · Peter A. Beling[1]

Maximum Margin Methods

- In a more general case, we have demonstrations $\mathrm{T} = \{\mathrm{T}_1, ..., \mathrm{T}_M\}$ and the objective here is to **find a reward R that maximises the expected value $E\left[V^\pi(\boldsymbol{s}_0)\right]$**, for $\boldsymbol{s}_0$ belonging to an initial state distribution $\mathcal{D}$

- The optimisation problem in this case is

$$\begin{array}{ll} \max \sum_{j=1}^{D} & p\left(\hat{V}^{\pi^*}(\boldsymbol{s}_0) - \hat{V}^{\pi_j}(\boldsymbol{s}_0)\right) \\ \text{such that} & |w_k| \leq 1, \; 1 \leq k \leq N \end{array}$$

where $p(x)$ is defined as on the previous slide and $\{\pi_1, ..., \pi_D\}$ is a set of candidate policies

Reward Estimation Given Example Trajectories

Maximum Margin Methods

Artificial Intelligence Review (2022) 55:4307–4346
https://doi.org/10.1007/s10462-021-10108-x
A survey of inverse reinforcement learning
Stephen Adams[1] · Tyler Cody[1] · Peter A. Beling[1]

- In a more general case, we have demonstrations $\mathrm{T} = \{\mathrm{T}_1, ..., \mathrm{T}_M\}$ and the objective here is to **find a reward R that maximises the expected value $E\left[V^{\pi}(s_0)\right]$**, for $\boldsymbol{s}_0$ belonging to an initial state distribution $\mathcal{D}$

- The optimisation problem in this case is

$$\begin{aligned} & \max \textstyle\sum_{j=1}^{D} \quad p\left(\hat{V}^{\pi^*}(\boldsymbol{s}_0) - \hat{V}^{\pi_j}(\boldsymbol{s}_0)\right) \\ & \text{such that} \quad |w_k| \leq 1, \ 1 \leq k \leq N \end{aligned}$$

  where $p(x)$ is defined as on the previous slide and $\{\pi_1, ..., \pi_D\}$ is a set of candidate policies

- This is **solved in an iterative fashion**:
  1. **The optimisation problem above is solved** to find a new reward function R_D
  2. **A new policy π_{D+1} is found using R_D** and the process is repeated

Artificial Intelligence Review (2022) 55:4307–4346
https://doi.org/10.1007/s10462-021-10108-x
A survey of inverse reinforcement learning
Stephen Adams[1] · Tyler Cody[1] · Peter A. Beling[1]

Reward Estimation by Expected Feature Maximisation

Maximum Margin Methods

- In a related formulation that is applicable to the case in which demonstrations T are given, the **expected feature values in the demonstrations** $\mu_E\left(\phi_k(\boldsymbol{s})\right)$ are of interest

Reward Estimation by Expected Feature Maximisation

Maximum Margin Methods

- In a related formulation that is applicable to the case in which demonstrations T are given, the **expected feature values in the demonstrations** $\mu_E\left(\phi_k(\boldsymbol{s})\right)$ are of interest
- The objective is to **find a reward function R that induces a policy π^* which minimises** $\|\mu_{\pi^*} - \mu_E\|$

Reward Estimation by Expected Feature Maximisation

Maximum Margin Methods

Artificial Intelligence Review (2022) 55:4307–4346
https://doi.org/10.1007/s10462-021-10108-x

A survey of inverse reinforcement learning

Stephen Adams[1] · Tyler Cody[1] · Peter A. Beling[1]

- In a related formulation that is applicable to the case in which demonstrations T are given, the **expected feature values in the demonstrations** $\mu_E\left(\phi_k(\boldsymbol{s})\right)$ are of interest
- The objective is to **find a reward function** R **that induces a policy** π^* **which minimises** $\|\mu_{\pi^*} - \mu_E\|$
- Here, the optimisation problem is written as

$$
\begin{aligned}
\max_{\boldsymbol{w},t} \quad & t \\
\text{such that} \quad & \textstyle\sum_{k=1}^{N} w_k \mu_E\left(\phi_k(\boldsymbol{s})\right) \leq \sum_{k=1}^{N} w_k \mu_{\pi_j}\left(\phi_k(\boldsymbol{s})\right) + t, \ 0 \leq j \leq D \\
& \|\boldsymbol{w}\| \leq 1
\end{aligned}
$$

Reward Estimation by Expected Feature Maximisation

Maximum Margin Methods

Artificial Intelligence Review (2022) 55:4307–4346
https://doi.org/10.1007/s10462-021-10108-x
A survey of inverse reinforcement learning
Stephen Adams[1] · Tyler Cody[1] · Peter A. Beling[1]

- In a related formulation that is applicable to the case in which demonstrations T are given, the **expected feature values in the demonstrations** $\mu_E\left(\phi_k(\boldsymbol{s})\right)$ are of interest
- The objective is to **find a reward function** R **that induces a policy** π^* **which minimises** $\|\mu_{\pi^*} - \mu_E\|$
- Here, the optimisation problem is written as

$$\begin{aligned} \max_{\boldsymbol{w},t} \quad & t \\ \text{such that} \quad & \textstyle\sum_{k=1}^{N} w_k \mu_E\left(\phi_k(\boldsymbol{s})\right) \leq \sum_{k=1}^{N} w_k \mu_{\pi_j}\left(\phi_k(\boldsymbol{s})\right) + t, \; 0 \leq j \leq D \\ & \|\boldsymbol{w}\| \leq 1 \end{aligned}$$

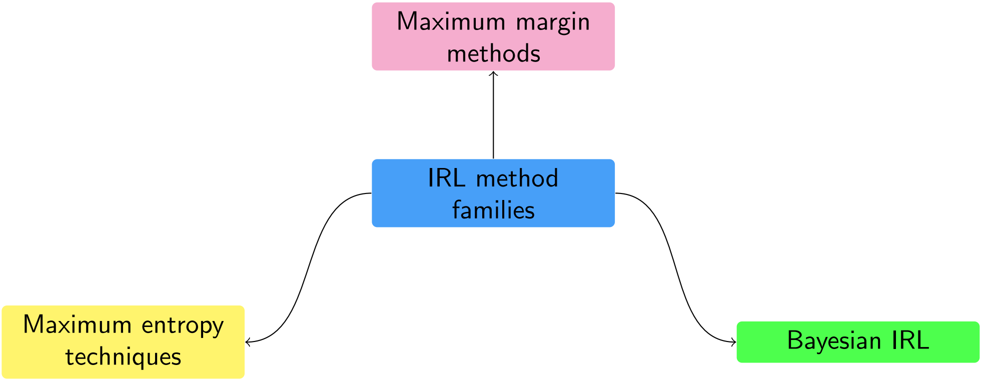
- Just as the problem on the previous slide, this problem is **solved iteratively**, with a new policy added to the candidate set at each iteration of the algorithm

IRL Methods Overview

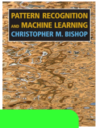

Artificial Intelligence Review (2022) 55:4307–4346
https://doi.org/10.1007/s10462-021-10108-x

A survey of inverse reinforcement learning

Stephen Adams[1] · Tyler Cody[1] · Peter A. Beling[1]



Reminder: Entropy, Maximum Entropy



- Given a random variable X, its entropy is defined as

Discrete X

$$H(X) = -\sum_{x_i \in X} P(x_i) \ln P(x_i)$$

Continuous X

$$H(X) = -\int p(x) \ln p(x) dx$$

- To maximise the entropy, H is optimised under the constraint that $\int p(x)dx = 1$:

Discrete X

$$H^* = -\sum_{x_i} P(x_i) \ln P(x_i) + \lambda \left(\sum_{x_i} P(x_i) - 1 \right)$$

Continuous X

$$H^* = -\int p(x) \ln p(x) dx + \lambda \left(\int p(x) dx - 1 \right)$$

- In maximum entropy IRL, an important result is the maximisation of H under an additional constraint of a known expectation of a function $f(x)$ (a result due to E. T. Jaynes, "Information theory and statistical mechanics," *Physical review*, vol. 106, no. 4, pp. 620–630, 1957.):

$$H^* = -\int p(x) \ln p(x) dx + \lambda \left(\int p(x) dx - 1 \right) + \nu \left(\sum_{i=1}^{N} p_i f(x_i) \right)$$

Maximum Entropy Inverse Reinforcement Learning

Artificial Intelligence Review (2022) 55:4307–4346
https://doi.org/10.1007/s10462-021-10108-x

A survey of inverse reinforcement learning

Stephen Adams[1] · Tyler Cody[1] · Peter A. Beling[1]

- As the name implies, maximum entropy techniques have the objective of finding **a reward that maximises the entropy** over a variable of interest, such as the feature counts that are present in the demonstrations
- In other words, maximum entropy IRL models **a distribution over the given demonstrations**, such that it looks for a distribution that maximises the entropy
- The earliest work on maximum entropy IRL is due to B. Ziebart et al. "Maximum Entropy Inverse Reinforcement Learning," in *Proc. 23rd AAAI Conf. Artificial Intelligence (AAAI)*, 2008, pp. 1433–1438.

Maximum Entropy IRL Formulation

Artificial Intelligence Review (2022) 55:4307–4346
https://doi.org/10.1007/s10462-021-10108-x

A survey of inverse reinforcement learning

Stephen Adams[1] · Tyler Cody[1] · Peter A. Beling[1]

- In maximum entropy IRL (but also in some maximum margin methods), a commonly used value is the **feature count along a trajectory**:

$$\boldsymbol{f}_{\mathrm{T}_i} = \sum_{\boldsymbol{s}_t \in \mathrm{T}_i} \phi(\boldsymbol{s}_t)$$

such that the full reward along T_i is $r(\boldsymbol{f}_{\mathrm{T}_i}) = \boldsymbol{\theta}^T \boldsymbol{f}_{\mathrm{T}_i}$, where $\boldsymbol{\theta}$ are weights

Artificial Intelligence Review (2022) 55:4307–4346
https://doi.org/10.1007/s10462-021-10108-x

A survey of inverse reinforcement learning

Stephen Adams[1] · Tyler Cody[1] · Peter A. Beling[1]

Maximum Entropy IRL Formulation

- In maximum entropy IRL (but also in some maximum margin methods), a commonly used value is the **feature count along a trajectory**:

$$\boldsymbol{f}_{\mathrm{T}_i} = \sum_{\boldsymbol{s}_t \in \mathrm{T}_i} \phi(\boldsymbol{s}_t)$$

  such that the full reward along T_i is $r(\boldsymbol{f}_{\mathrm{T}_i}) = \boldsymbol{\theta}^T \boldsymbol{f}_{\mathrm{T}_i}$, where $\boldsymbol{\theta}$ are weights

- The probability of all trajectories T is then modelled as

$$P\left(\mathrm{T}|\theta\right) = \frac{1}{Z} e^{\boldsymbol{\theta}^T \boldsymbol{f}_{\mathrm{T}}} \prod_{\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1}} P(\boldsymbol{s}_{t+1} | \boldsymbol{s}_t, \boldsymbol{a}_t) = \frac{1}{Z} e^{\sum_{i=1}^{M} \boldsymbol{\theta}^T \boldsymbol{f}_{\mathrm{T}_i}} \prod_{\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1}} P(\boldsymbol{s}_{t+1} | \boldsymbol{s}_t, \boldsymbol{a}_t)$$

Maximum Entropy IRL Formulation


Artificial Intelligence Review (2022) 55:4307–4346
https://doi.org/10.1007/s10462-021-10108-x
A survey of inverse reinforcement learning
Stephen Adams[1] · Tyler Cody[1] · Peter A. Beling[1]

- In maximum entropy IRL (but also in some maximum margin methods), a commonly used value is the **feature count along a trajectory**:

$$\boldsymbol{f}_{\mathrm{T}_i} = \sum_{\boldsymbol{s}_t \in \mathrm{T}_i} \phi(\boldsymbol{s}_t)$$

  such that the full reward along T_i is $r(\boldsymbol{f}_{\mathrm{T}_i}) = \boldsymbol{\theta}^T \boldsymbol{f}_{\mathrm{T}_i}$, where $\boldsymbol{\theta}$ are weights

- The probability of all trajectories T is then modelled as

$$P\left(\mathrm{T}|\theta\right) = \frac{1}{Z} e^{\boldsymbol{\theta}^T \boldsymbol{f}_{\mathrm{T}}} \prod_{\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1}} P(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t) = \frac{1}{Z} e^{\sum_{i=1}^{M} \boldsymbol{\theta}^T \boldsymbol{f}_{\mathrm{T}_i}} \prod_{\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1}} P(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t)$$

- The optimal weights (which maximise the entropy) are found by **maximum likelihood estimation**:

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{M} \log\ P(\mathrm{T}_i|\boldsymbol{\theta})$$


Artificial Intelligence Review (2022) 55:4307–4346
https://doi.org/10.1007/s10462-021-10108-x
A survey of inverse reinforcement learning
Stephen Adams[1] · Tyler Cody[1] · Peter A. Beling[1]

Maximum Entropy IRL Formulation

- In maximum entropy IRL (but also in some maximum margin methods), a commonly used value is the **feature count along a trajectory**:

$$\boldsymbol{f}_{\mathrm{T}_i} = \sum_{\boldsymbol{s}_t \in \mathrm{T}_i} \phi(\boldsymbol{s}_t)$$

  such that the full reward along T_i is $r(\boldsymbol{f}_{\mathrm{T}_i}) = \boldsymbol{\theta}^T \boldsymbol{f}_{\mathrm{T}_i}$, where $\boldsymbol{\theta}$ are weights

- The probability of all trajectories T is then modelled as

$$P\left(\mathrm{T}|\theta\right) = \frac{1}{Z} e^{\boldsymbol{\theta}^T \boldsymbol{f}_{\mathrm{T}}} \prod_{\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1}} P(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t) = \frac{1}{Z} e^{\sum_{i=1}^{M} \boldsymbol{\theta}^T \boldsymbol{f}_{\mathrm{T}_i}} \prod_{\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1}} P(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t)$$

- The optimal weights (which maximise the entropy) are found by **maximum likelihood estimation**:

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{M} \log\ P(\mathrm{T}_i|\boldsymbol{\theta})$$

- The function is maximised using gradient ascent, such that the gradient is the difference between the observed feature counts $\tilde{\boldsymbol{f}}$ and the expected feature counts:

$$\nabla L(\boldsymbol{\theta}) = \tilde{\boldsymbol{f}} - \sum_{i=1}^{M} P(\mathrm{T}_i|\boldsymbol{\theta}) \boldsymbol{f}_{\mathrm{T}_i}$$

Maximum Entropy IRL Methods Overview

Artificial Intelligence Review (2022) 55:4307–4346
https://doi.org/10.1007/s10462-021-10108-x

A survey of inverse reinforcement learning

Stephen Adams[1] · Tyler Cody[1] · Peter A. Beling[1]

- On the previous slide, we looked at the original formulation from Ziebart et al. (2008), which is applicable in discrete state spaces and assumes optimal demonstrations

Maximum Entropy IRL Methods Overview

Artificial Intelligence Review (2022) 55:4307–4346
https://doi.org/10.1007/s10462-021-10108-x

A survey of inverse reinforcement learning

Stephen Adams[1] · Tyler Cody[1] · Peter A. Beling[1]

- On the previous slide, we looked at the original formulation from Ziebart et al. (2008), which is applicable in discrete state spaces and assumes optimal demonstrations
- Extensions of maximum entropy IRL exist that consider **suboptimal demonstrations**
  - One way to achieve this is to include an additional optimisation constraint that **maximises the difference to known suboptimal demonstrations**

Maximum Entropy IRL Methods Overview

Artificial Intelligence Review (2022) 55:4307–4346
https://doi.org/10.1007/s10462-021-10108-x

A survey of inverse reinforcement learning

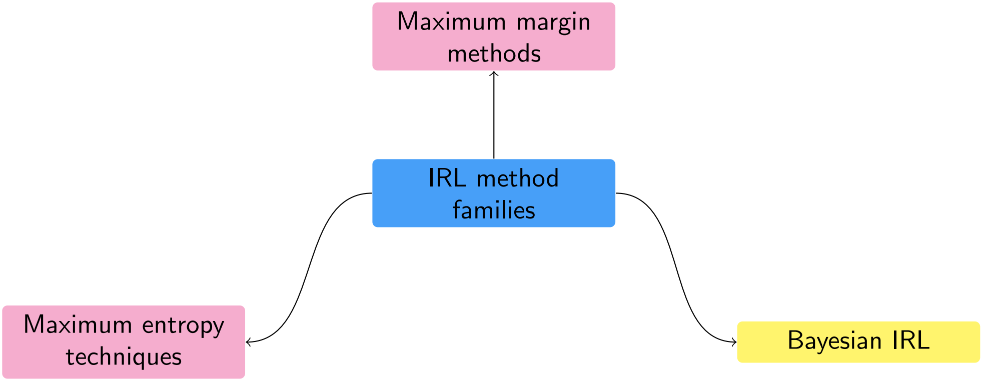
Stephen Adams[1] · Tyler Cody[1] · Peter A. Beling[1]

- On the previous slide, we looked at the original formulation from Ziebart et al. (2008), which is applicable in discrete state spaces and assumes optimal demonstrations
- Extensions of maximum entropy IRL exist that consider **suboptimal demonstrations**
  - One way to achieve this is to include an additional optimisation constraint that **maximises the difference to known suboptimal demonstrations**
- Extensions over **continuous state spaces** exist as well, for instance using a neural network as a reward estimator

IRL Methods Overview

Artificial Intelligence Review (2022) 55:4307–4346
https://doi.org/10.1007/s10462-021-10108-x

A survey of inverse reinforcement learning
Stephen Adams[1] · Tyler Cody[1] · Peter A. Beling[1]



Bayesian Inverse Reinforcement Learning

Artificial Intelligence Review (2022) 55:4307–4346
https://doi.org/10.1007/s10462-021-10108-x

A survey of inverse reinforcement learning

Stephen Adams[1] · Tyler Cody[1] · Peter A. Beling[1]

- The idea behind Bayesian IRL follows the general idea of Bayesian reasoning — **prior information** about a reward function is combined with **evidence provided by expert demonstrations** to obtain **an improved reward function estimate**

Bayesian Inverse Reinforcement Learning

Artificial Intelligence Review (2022) 55:4307–4346
https://doi.org/10.1007/s10462-021-10108-x

A survey of inverse reinforcement learning

Stephen Adams[1] · Tyler Cody[1] · Peter A. Beling[1]

- The idea behind Bayesian IRL follows the general idea of Bayesian reasoning — **prior information** about a reward function is combined with **evidence provided by expert demonstrations** to obtain **an improved reward function estimate**
- Unlike max margin methods, which obtain a point estimate of the reward function, Bayesian IRL results in **a distribution of rewards**
  - As a result, this family of methods can be applied even if the demonstrations are not optimal, just as maximum entropy IRL

Bayesian Inverse Reinforcement Learning

Artificial Intelligence Review (2022) 55:4307–4346
https://doi.org/10.1007/s10462-021-10108-x

A survey of inverse reinforcement learning

Stephen Adams[1] · Tyler Cody[1] · Peter A. Beling[1]

- The idea behind Bayesian IRL follows the general idea of Bayesian reasoning — **prior information** about a reward function is combined with **evidence provided by expert demonstrations** to obtain **an improved reward function estimate**
- Unlike max margin methods, which obtain a point estimate of the reward function, Bayesian IRL results in **a distribution of rewards**
  - As a result, this family of methods can be applied even if the demonstrations are not optimal, just as maximum entropy IRL
- The incorporation of prior knowledge can also **reduce the amount of demonstrations necessary for learning a sufficiently good reward estimate**
- The first Bayesian IRL method is due to D. Ramachandran and E. Amir, "Bayesian Inverse Reinforcement Learning," in *Proc. Int. Joint. Conf. Artificial Intelligence (IJCAI)*, vol. 7, 2007, pp. 2586–2591.

Bayesian IRL Formulation

Artificial Intelligence Review (2022) 55:4307–4346
https://doi.org/10.1007/s10462-021-10108-x
A survey of inverse reinforcement learning
Stephen Adams[1] · Tyler Cody[1] · Peter A. Beling[1]

- The probability of a trajectory given a reward is calculated under an **independence assumption**:

$$P(\mathrm{T}_i|R) = P\left((\boldsymbol{s}_1, \boldsymbol{a}_1)|R\right) \cdots P\left((\boldsymbol{s}_{\mathrm{T}_i^m}, \boldsymbol{a}_{\mathrm{T}_i^m})|R\right)$$

Artificial Intelligence Review (2022) 55:4307–4346
https://doi.org/10.1007/s10462-021-10108-x
A survey of inverse reinforcement learning
Stephen Adams[1] · Tyler Cody[1] · Peter A. Beling[1]

Bayesian IRL Formulation

- The probability of a trajectory given a reward is calculated under an **independence assumption**:

$$P(\mathrm{T}_i|R) = P\left((\boldsymbol{s}_1, \boldsymbol{a}_1)|R\right) \cdots P\left((\boldsymbol{s}_{\mathrm{T}_i^m}, \boldsymbol{a}_{\mathrm{T}_i^m})|R\right)$$

- The probability of a state-action pair given a reward is modelled as **an exponential function of the reward-specific state-action value function**:

$$P\left((\boldsymbol{s}_t, \boldsymbol{a}_t)|R\right) = \frac{1}{Z} e^{\alpha Q^*(\boldsymbol{s}_t, \boldsymbol{a}_t, R)}$$

Artificial Intelligence Review (2022) 55:4307–4346
https://doi.org/10.1007/s10462-021-10108-x
A survey of inverse reinforcement learning
Stephen Adams[1] · Tyler Cody[1] · Peter A. Beling[1]

Bayesian IRL Formulation

- The probability of a trajectory given a reward is calculated under an **independence assumption**:

$$P(\mathrm{T}_i|R) = P\left((\boldsymbol{s}_1, \boldsymbol{a}_1)|R\right) \cdots P\left((\boldsymbol{s}_{\mathrm{T}_i^m}, \boldsymbol{a}_{\mathrm{T}_i^m})|R\right)$$

- The probability of a state-action pair given a reward is modelled as **an exponential function of the reward-specific state-action value function**:

$$P\left((\boldsymbol{s}_t, \boldsymbol{a}_t)|R\right) = \frac{1}{Z} e^{\alpha Q^*(\boldsymbol{s}_t, \boldsymbol{a}_t, R)}$$

- Under this model, the probability of a trajectory is given as

$$P(\mathrm{T}_i|R) = \frac{1}{Z} e^{\alpha \sum_t Q^*(\boldsymbol{s}_t, \boldsymbol{a}_t, R)}$$

Bayesian IRL Formulation

Artificial Intelligence Review (2022) 55:4307–4346
https://doi.org/10.1007/s10462-021-10108-x

A survey of inverse reinforcement learning

Stephen Adams[1] · Tyler Cody[1] · Peter A. Beling[1]

- The probability of a trajectory given a reward is calculated under an **independence assumption**:

$$P(\mathrm{T}_i|R) = P\left((\boldsymbol{s}_1, \boldsymbol{a}_1)|R\right) \cdots P\left((\boldsymbol{s}_{\mathrm{T}_i^m}, \boldsymbol{a}_{\mathrm{T}_i^m})|R\right)$$

- The probability of a state-action pair given a reward is modelled as **an exponential function of the reward-specific state-action value function**:

$$P\left((\boldsymbol{s}_t, \boldsymbol{a}_t)|R\right) = \frac{1}{Z} e^{\alpha Q^*(\boldsymbol{s}_t, \boldsymbol{a}_t, R)}$$

- Under this model, the probability of a trajectory is given as

$$P(\mathrm{T}_i|R) = \frac{1}{Z} e^{\alpha \sum_t Q^*(\boldsymbol{s}_t, \boldsymbol{a}_t, R)}$$

- The posterior of the reward given a trajectory is thus:

$$\begin{aligned} P(R|\mathrm{T}_i) &= \eta P(\mathrm{T}_i|R) P(R) \\ &= \eta' e^{\alpha \sum_t Q^*(\boldsymbol{s}_t, \boldsymbol{a}_t, R)} P(R) \end{aligned}$$

Bayesian IRL Formulation

Artificial Intelligence Review (2022) 55:4307–4346
https://doi.org/10.1007/s10462-021-10108-x
A survey of inverse reinforcement learning
Stephen Adams[1] · Tyler Cody[1] · Peter A. Beling[1]

- The probability of a trajectory given a reward is calculated under an **independence assumption**:

$$P(\mathrm{T}_i|R) = P\left((\boldsymbol{s}_1, \boldsymbol{a}_1)|R\right) \cdots P\left((\boldsymbol{s}_{\mathrm{T}_i^m}, \boldsymbol{a}_{\mathrm{T}_i^m})|R\right)$$

- The probability of a state-action pair given a reward is modelled as **an exponential function of the reward-specific state-action value function**:

$$P\left((\boldsymbol{s}_t, \boldsymbol{a}_t)|R\right) = \frac{1}{Z} e^{\alpha Q^*(\boldsymbol{s}_t, \boldsymbol{a}_t, R)}$$

- Under this model, the probability of a trajectory is given as

$$P(\mathrm{T}_i|R) = \frac{1}{Z} e^{\alpha \sum_t Q^*(\boldsymbol{s}_t, \boldsymbol{a}_t, R)}$$

- The posterior of the reward given a trajectory is thus:

$$\begin{aligned} P(R|\mathrm{T}_i) &= \eta P(\mathrm{T}_i|R) P(R) \\ &= \eta' e^{\alpha \sum_t Q^*(\boldsymbol{s}_t, \boldsymbol{a}_t, R)} P(R) \end{aligned}$$

- Given the posterior distribution, the **mean** or the **maximum a posteriori (MAP)** estimate can be used as an estimate of the reward

Artificial Intelligence Review (2022) 55:4307–4346
https://doi.org/10.1007/s10462-021-10108-x

A survey of inverse reinforcement learning

Stephen Adams[1] · Tyler Cody[1] · Peter A. Beling[1]

Bayesian IRL Methods Overview

- The formulation on the previous slide is the original model from Ramachandran and Amir (2007), but there is a large variety of Bayesian IRL algorithms that differ along multiple dimensions:

Bayesian IRL Methods Overview

Artificial Intelligence Review (2022) 55:4307–4346
https://doi.org/10.1007/s10462-021-10108-x

A survey of inverse reinforcement learning
Stephen Adams[1] · Tyler Cody[1] · Peter A. Beling[1]

- The formulation on the previous slide is the original model from Ramachandran and Amir (2007), but there is a large variety of Bayesian IRL algorithms that differ along multiple dimensions:
  - **The estimation of the mean or MAP estimate** (sampling from the posterior is usually done for this purpose, or the prior is written in a form that leads to a tractable posterior, such as a Gaussian distribution)

Bayesian IRL Methods Overview


Artificial Intelligence Review (2022) 55:4307–4346
https://doi.org/10.1007/s10462-021-10108-x

A survey of inverse reinforcement learning

Stephen Adams[1] · Tyler Cody[1] · Peter A. Beling[1]

- The formulation on the previous slide is the original model from Ramachandran and Amir (2007), but there is a large variety of Bayesian IRL algorithms that differ along multiple dimensions:
  - **The estimation of the mean or MAP estimate** (sampling from the posterior is usually done for this purpose, or the prior is written in a form that leads to a tractable posterior, such as a Gaussian distribution)
  - **Learning from multiple demonstrators that may follow different rewards** (clustering is sometimes used for this purpose)

Bayesian IRL Methods Overview

Artificial Intelligence Review (2022) 55:4307–4346
https://doi.org/10.1007/s10462-021-10108-x
A survey of inverse reinforcement learning
Stephen Adams[1] · Tyler Cody[1] · Peter A. Beling[1]

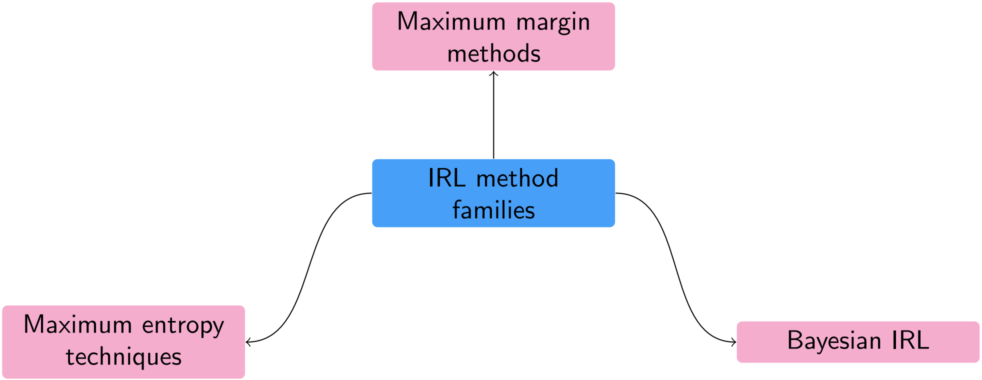
- The formulation on the previous slide is the original model from Ramachandran and Amir (2007), but there is a large variety of Bayesian IRL algorithms that differ along multiple dimensions:
  - **The estimation of the mean or MAP estimate** (sampling from the posterior is usually done for this purpose, or the prior is written in a form that leads to a tractable posterior, such as a Gaussian distribution)
  - **Learning from multiple demonstrators that may follow different rewards** (clustering is sometimes used for this purpose)
  - **Reward learning that incorporates negative demonstrations** rather than only positive examples, as well as **learning to account for states that are not visited during the demonstrations** (Gaussian processes can be used to model the posterior in this case)

IRL Methods Overview

Artificial Intelligence Review (2022) 55:4307–4346
https://doi.org/10.1007/s10462-021-10108-x

A survey of inverse reinforcement learning

Stephen Adams[1] · Tyler Cody[1] · Peter A. Beling[1]



Summary

- For learning robot policies (using reinforcement learning), a reward function is typically needed, but it is often challenging to write down a good enough reward function manually
- Inverse reinforcement learning (IRL) is the problem of learning a reward function, typically given a set of expert demonstrations
- There are multiple classes of IRL methods; we particularly looked at maximum margin methods, maximum entropy IRL, and Bayesian IRL