Intelligent Environments 2021
E. Bashir and M. Luštrek (Eds.)

doi:10.3233/AISE210097

An Overview of Inverse Reinforcement Learning Techniques

Syed Ihtesham Hussain Shah [a,b,1], Giuseppe De Pietro [b]
[a] *Dept. of ICT and Engineering, University of Parthenope, Italy*
[b] *Institute for high performance computing and networking (ICAR), National Research Council, Italy*

Abstract. In decision-making problems reward function plays an important role in finding the best policy. Reinforcement Learning (RL) provides a solution for decision-making problems under uncertainty in an Intelligent Environment (IE). However, it is difficult to specify the reward function for RL agents in large and complex problems. To counter these problems an extension of RL problem named Inverse Reinforcement Learning (IRL) is introduced, where reward function is learned from expert demonstrations. IRL is appealing for its potential use to build autonomous agents, capable of modeling others, deprived of compromising in performance of the task. This approach of learning by demonstrations relies on the framework of Markov Decision Process (MDP). This article elaborates original IRL algorithms along with their close variants to mitigate challenges. The purpose of this paper is to highlight an overview and theoretical background of IRL in the field of Machine Learning (ML) and Artificial Intelligence (AI). We presented a brief comparison between different variants of IRL in this article.

Keywords. Inverse Reinforcement Learning, Markov Decision Process, Intelligent Environment

Introduction

ML is an application of AI that focuses on learning and improving itself from experience and without being explicitly programmed. ML emphasizes on developing algorithms that can access data and use it for self-learning [1,2,3] in an intelligent environments [4,5,6]. We are dealing with a certain number of sensors, which enable the IE [7] to be aware of the user's current action and goal. Human activities are observable through different sensors [8] and observations can be assumed to teach another environmental device or system to perform the task in a better way [9]. A typical way of teaching a system in a decision-making problem requires direct coding. However, another paradigm called Learning from Demonstrations has emerged to teach devices via demonstrations [10].
RL techniques provide a solution for decision-making problems [11]. RL agent interact with the dynamic environment, gains experience and improve itself [12]. The result of an agent's interaction is a policy that can provide solutions to complex tasks without having

[1]Corresponding Author: Syed Ihtesham Hussain Shah, Institute for high performance computing and networking (ICAR), National Research Council (CNR), Via Pietro Castellino, 111 Napoli – 80131, Italy. E-mail: ihtesham.shah@icar.cnr.it

specific information about the underlying problem [13]. Unlike supervised learning, RL does not need output labels that are sometimes not available or maybe expensive to find in real-time application. RL has better generalization abilities and can easily be applied to more complex scenarios [14] e.g. minimize medication errors [15], [16] and risk management [17].
However, the problem associated with RL is the reward function that has to be specified in advance. For complex and large problems, it is very difficult to specify and exhaustive to tune the reward function. To counter design difficulties of RL, IRL is introduced. IRL is an extension of RL problem where reward function is learned through expert demonstrations. In this paper, we present fundamental and advanced techniques of IRL [18] for finding the best-fitting reward function for expert trajectories.
The rest of the paper is organized as follows: Section-1 comprises a brief review of the background and problem formulation, where basics about the MDP is presented. Original IRL algorithm is discussed in section-2. This section describes the problem of learning the reward function not explicitly, but through observing an expert demonstration. Section-3 introduces an overview of different variants and extensions of IRL in emerging and existing fields. We summarize the paper in section-4 by giving a conclusion.

1. Theoretical Background

1.1. Markov Decision Process

Generally, RL algorithms satisfy Markov Decision Process (MDP) that are based on Markov property. It does not consider past information while taking actions in current state. MDP is a form of tuple (S, A, T, R, γ) [19] explained below:

- $S = s_1, s_2, s_3, ...s_n$ represent set of all possible states in the given environment. Based upon the chosen action and transition function the agent can move to any of these states.
- $A = a_1, a_2, a_3, ...a_n$ is a set of all possible actions that an agent can take in a state at every time step.
- $T(s_t, a_t, s_(t+1))$ is a transition function which tells the probability of having into the state $s_(t+1)$ by taking action a in current state s in time step t.
- $R(s_t, a_t, s_(t+1))$ is a reward function which tells about the cost of taking action a in state s in time step t.
- $\gamma \in [0, 1]$: is a discount factor that takes the value between zero and one. A value close to zero gives more weight to current rewards while a value close to one gives more weight to long term rewards.

1.2. Value Function

A policy π is a function of mapping state s to an action a. It can be stochastic $\pi : S \rightarrow Prob(A)$ or deterministic $\pi : S \rightarrow A$ [20]. For a policy π, V^π is the value function which estimates the accumulated reward value in state s_t and follow the policy π. The value of a policy V^π for a given state s is given as:

$$V^{\pi}(s) = E_{\pi}\{R_t|s_t = s\} = \sum_{a\in A(s)} \pi(s,a) \sum_{s_{t+1}\in S} T^{a}_{ss_{t+1}}\{r(s,a) + \gamma V^{\pi}(s_{t+1})\} \tag{1}$$

Where V^{π} is the value function to estimate the accumulated reward and R_t is the reward function which is calculated as given below.

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots\ldots \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \tag{2}$$

Q function is the expected aggregated return when you take action a in state s.

$$Q^{\pi}(s,a) = \sum_{\pi}\{R_t|s_t = s, a_t = a\} \tag{3}$$

The goal of solving MDP is to find optimal policy π^*. The state-action value function (Q) or state value function (V) for optimal policy π^* can be calculated by using Bellman equations as given below:

$$V^*(s) = \max_a \sum_{s_{t+1}\in S} T^{a}_{ss_{t+1}}(R^{a}_{ss_{t+1}} + \gamma V^*(s_{t+1})) \tag{4}$$

$$Q^*(s,a) = \sum_{s_{t+1}\in S} T^{a}_{ss_{t+1}}(R^{a}_{ss_{t+1}} + \gamma \max_{at+1} V^*(s_{t+1}, a_{t+1})) \tag{5}$$

Eqs. (4) and (5) explains the objective of learning the policy that provides the maximum reward return. Other machine learning techniques have successfully been applied to many real-world problems i.e. skill transfer to robots [8] and autonomous navigation [7] etc. However, these techniques are usually suffering from fewer training samples and poor generalization. The conventional terminologies of MDP and RL are usually accepting fix and predetermined reward functions. However, it is difficult to specify appropriate reward functions particularly for complex and large problems [10]. Therefore, the researcher gave a solution in form of IRL to tackle these limitations. IRL methods flip the RL problem and rather than finding optimal policy it tries to find underlying rewards for some given policies.

2. Inverse Reinforcement Learning (IRL)

Generally, IRL supposes that the expert is acting according to an underlying policy π_E. In some cases, the policy may not be known, and learners observe sequences of the expert's state-action pairs called trajectories. It follows some composition of a model that helps in learning the unknown rewards function for these trajectories. Conventional models include, a linearly weighted combination of reward features, non-linear (probability distribution over reward functions) or a neural network representation [21].
Typical framework of IRL is shown in Figure-1. Expert trajectories are supposed to generate optimal policy. Rather than the policy, the reward function is the briefest, robust,

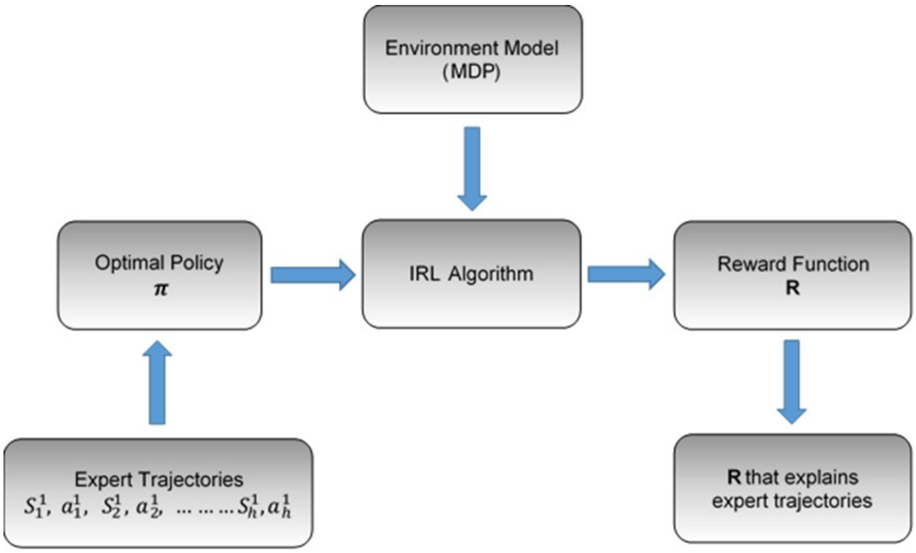
Environment Model (MDP)
Optimal Policy π
IRL Algorithm
Reward Function R
Expert Trajectories $S_1^1, a_1^1, S_2^1, a_2^1, \dots\dots\dots S_h^1, a_h^1$
R that explains expert trajectories

Figure 1. Inverse Reinforcement Learning Problem

and convenient definition of the task. It computes how bad or good certain actions are. Therefore, the goal here is to find the reward function that is being implicitly optimized by the optimal policy π^*. Once we have the true reward function, the problem is concentrated on finding the right policy, and standard reinforcement learning methods can be applied to solve the problem.
IRL as an optimization problem and proposed Linear Programming (LP) approach to solve the problem by considering three types of state spaces [22].

2.1. *LP for finite state space*

First, consider the set of optimal policies π for a MDP with a finite state space S and policy transition matrix $T^{\neg\pi}_{n*n}$. For each state only one action provide an optimal solution from set of K possible actions and all other $k-1$ actions are non-policy actions hence non policy transition matrix are $T^{\neg\pi} = T^1, \dots T^{k-1}$. Action is considered optimal only when the reward function satisfy the following equation:

$$(T^{\neg\pi} - T^{\pi})(I - \gamma T^{\pi})^{-1} R0 \tag{6}$$

Where $T^{\neg\pi}$ is non policy transition matrix and T^{π} represents policy transition matrix. Many solutions satisfy Eq. (6). let say assigning zero reward value to all the states is always a solution. To remove this problem two methods are proposed.

Costly single-step deviation. It maximizes the distance between the optimal policy Q value and the second-best value among all the others.

$$\textbf{maximize}: \Sigma_{s \varepsilon S} Q^{\pi}(s, \pi(s)) - \max_{a \varepsilon A|\pi(s)} Q^{\pi}(s,a) \tag{7}$$

Eq. (7) successfully removes several degenerate reward functions but the reward function that is learned by this kind of heuristics might be different from the real one. To tackle this issue, a second method is introduced which assumes that reward function with **many small rewards** are more natural and should be preferred. Maximization of many terms leads to a minimization of the reward vector's norm i.e. $l_1 - norm$: $-\lambda \parallel \hat{R} \parallel_1$. A suitable value for parameter λ can be searched automatically. In addition, an upper bound value for reward function is also imposed.

$$\forall (s \, \varepsilon \, S) \quad | \, \hat{R} \, | \leq \; R_{max} \tag{8}$$

However, this approach cannot be applied to very large state spaces.

2.2. *LP for Infinite State Space*

For this kind of state-space researcher have used the function approximation to find a linear combination of d known, fixed and bounded basis function Φ_i [23].

$$\hat{R}(s) \, = \, \alpha_1 \, \phi_1(s) \, + \, \alpha_2 \, \phi_2(s) \ldots\ldots + \, \alpha_d \, \phi_d \tag{9}$$

Where $\alpha \rightarrow [0,1]$ is a constant. However, this linear combination comes with a huge disadvantage. We are not sure that the estimated reward function is the one with true value. Eq. (9) only expresses a smaller subset of all the possible reward functions. On the other hand, V^{π} can be expressed in term of basis function as follows:

$$V^{\pi} \, = \, \alpha_1 \, V_1^{\pi} \, + \ldots\ldots + \, \alpha_d \, V_d^{\pi} \tag{10}$$

Transition probabilities can't be expressed in the term matrix anymore because of infinite state space. Instead, the expected value based on sampled subset $s_0 \subset S$ is estimated and it is enforced that non-optimal actions always lead to lower expected value than optimal actions.

$$\forall_{s_0 \, \varepsilon \, S}, \forall_{a \, \varepsilon \, A} \textbf{:} \; E_{\acute{s} \; T(\acute{s}|s,\pi(s))} \{V^{\pi}(\acute{s})\} \, \geq \, E_{\acute{s} \; T(\acute{s}|s,a)} \{V^{\pi}(\acute{s})\} \tag{11}$$

A penalization factor p is introduced that define how much constraints can be penalized [18].

$$\max \sum_{s \in S_0} \min_{a \in A | \pi(s)} \mathrm{p} \, [E_{\acute{s} \; T(\acute{s}|s,\pi(s))} (V^{\pi}(\acute{s})) - (E_{\acute{s} \; T(\acute{s}|s,a)}) (V^{\pi}(\acute{s}))] \tag{12}$$

This formulation maximizes the smallest difference between the expected value generated by policy action and all the other expected values it could find.

2.3. *LP with Sampled Trajectories*

This third algorithm is more closer to the real-world problem. It deals with the scenario where the knowledge about the exact policy is not given. we only observe trajectories that are combinations of states and actions. Optimal policy based on some unknown reward function generates these trajectories.
However, the goal is to calculate the empirical value of a trajectory rather than the ex-

Algorithm 1 Generic Algorithm for IRL

```
1: **Given:** Expert demonstrations, discount factor $\gamma$, termination criteria $\varepsilon$,
2: **Initialize:** Feature matrix $\phi$, number of iteration $n$.
3: while (Termination criteria fulfil) do
4:     for $\forall\,(s,a)$ do
5:         Solve optimal value function $V^*$ for MDP.
6:         Use $V^*$ to define policy $\hat{\pi}$ .
7:         Choose parameter $\alpha_i$ to make $\hat{\pi}$ more similar to demonstration;
8:     end for
9: end while
10: Return: $R(\hat{s},a)$
```

pected value of the policy. By using current estimated reward function $\hat{R}$ corresponding value estimate $\hat{V}_i(\xi)$ for each value function can be calculated as given below:

$$\hat{V}_i(\xi) = \sum_{s_j \in \xi} \gamma^j \phi_i(s_j) \tag{13}$$

γ is a discount factor. Empirical value for overall trajectory is :

$$\hat{V}(\xi) = \sum_{i=1}^{d} \alpha_i \hat{V}_i(\xi), \quad |\alpha_i| \leq 1\,,\; i = \{1,....,d\} \tag{14}$$

There are might be many expert trajectories generated at different initial positions. Now the goal is to find such a value of $_i$ that yields a higher empirical reward for optimal trajectory.

$$\hat{V}(\xi_{\pi^*}) \geq \hat{V}(\xi_{\pi^i}) \tag{15}$$

Eq. (15) represent that optimal expert trajectory always generates higher empirical value $\hat{V}(\xi_{\pi^*})$ than any other policy. Linear programming is used to find the best fitting parameter α_i. There are two important assumption made here [23]:

1. For any given policy trajectories can be generated.
2. Given any reward function, a policy which is optimal for this reward function can be generated.

To find the true empirical value, limit the maximum value of factor α_i and a penalization factor p (as stated in the previous section)is also introduced.

$$maximize\,[\sum_{i=1}^{m} \mathrm{p}\,\{\hat{V}(\xi_{\pi^*}) - \hat{V}(\xi_{\pi^i})\}] \tag{16}$$

A generic algorithm for IRL is given in Algorithm-1. The key idea is to use observation from the expert policy and find information about the underlying MDP. Estimation of the optimal policy is performed through the execution of the algorithm iteratively.

3. Foundation of IRL Variants

Many changes have been made in the fundamental IRL algorithm depending upon the nature of the application. We will try to give a brief introduction to some of them in this section.

3.1. Maximum Margin Planning

Let consider an MDP without reward MDP/R. In this context we have the same component as regular MDP except for the reward model. Given some expert's feature expectation μ_E and feature mapping ϕ, the goal is to find a policy that explains expert behavior perfectly on an unknown reward function. Reward function might be a linear combination of features.

$$R(s,a) = w_1\phi_1(s,a) + w_2\phi_2(s,a) + \ldots\ldots + w_k\phi_k(s,a) \tag{17}$$

$$R(s,a) = \sum_i \{w_i\phi_i(s,a)\} \tag{18}$$

Where feature function $\phi_i : S \to R$ and weights $w_i \in R$. To find a policy $\hat{\pi}$ such that $||\mu\hat{\pi} - \mu_E||_2 \leq \varepsilon$ we define:

$$E[\sum_{t=1}^{\infty}{}^{t}R(s_t)|\pi_E] - E[\sum_{t=1}^{\infty}{}^{t}R(s_t)|\hat{\pi}] \;=\; ||W^t||\,||\mu_E - \mu(\hat{\pi})||_2 \tag{19}$$

where W : $(||w||_1 \leq 1)$. Maximum margin algorithm are used to find such a policy $\hat{\pi}$ that minimize the difference between expert feature expectation μ_E and other random chosen policy expectation $\mu\hat{\pi}$.

3.2. Maximum Entropy IRL

Maximum entropy uses the probability approach to resolve the ill-posed issue associated with original IRL. Maximum entropy [24] is based on the principle that makes it free from the arbitrary assumption about the missing information of the system. The trajectories of experts also weighted with the estimated rewards and preference given to those policies with high rewards [25].

$$P_x(O_x|\theta) = [1/Z_i]exp(\alpha_x E(O_x,\theta)) \tag{20}$$

In Eq. (20), θ represents the parameters of the reward function and R is a linear combination of features as discussed in the previous section. Through the maximum likelihood approach of observed trajectory, we can get the optimum value of θ [26].

$$\theta^* = \arg\max_{\theta} L(\theta) \tag{21}$$

$$\theta^* = \arg\max_{\theta} \sum_{o} \log P_x(O_x|\theta) \tag{22}$$

Gradient-based optimization method can be used to obtain the optimum values for deterministic MDP. The gradient is presented by the difference between expected feature values by the learner and expected empirical feature values. Early IRL based methods are countered with the problem of label bias mentioned in [27]. It only considers actions in run time while trajectories are compared later-on by taking the actions instead of comparing before. Consequently, the best policy having the highest reward may not be the one with the highest probability [28]. Maximum entropy encounters this problem by focusing on distribution over trajectories rather than actions. This algorithm applied in many applications i.e. to predict the driver behavior, route recommendation for taxi driver [29].

3.3. Non-linear Programming

Max-Margin Planning (MMP) still assumes a linear form of reward function but another approach LEArning and seaRCH (LEARCH) [30] have introduced for nonlinear behavior of reward function. This algorithm is tested in autonomous navigation where an agent (vehicle) was operated in complex unstructured terrain [31]. A parallel approach was also introduced and used in a visual navigation system where costs to detected objects and a suitable path to the drivers in the current situation have been assigned automatically [32]. Self-Imitation Learning (IL) [33] proposes that without any feedback from an external expert, the policy can be learned iteratively after the agent makes mistakes and decisions.

3.4. Maximum Likelihood IRL

Maximum Likelihood IRL (MLIRL) [34] problem utilizing an estimate of the gradient of the likelihood function. The algorithm defines that likelihood of the expert data-set $\mathscr{L}_\theta(O)$ can be represented by the product of the likelihood of the state-action pairs.

$$\mathscr{L}_\theta(O) = \prod_{i=1}^{k} l_\theta(s_i, a_i) \tag{23}$$

Where O represents experts demonstration. The reward function is estimated by Maximizing the log-likelihood function.

$$R^*_\theta = \arg\max_{R} \log \mathscr{L}_\theta(O) \tag{24}$$

Eq. (24) assumes that actions are more likely to be selected having higher Q^* values.

3.5. Gradient-based IRL

The neural gradient [29] approach was used to refine the reward function instead of directly modifying the policy. Policies are derived from the RL algorithm for the current reward function in MMP. Therefore, policy space depends on the reward function, and any change in the parameter space θ^* affects the policy space. So a neural network is used to solve θ^* by gradient approach. By using the gradient descent method, IRL stretched to several target settings [35]. In [27] an idea was introduced where any deviation from the expert's trajectory was corrected.

3.6. Monte-Carlo Markov Chain (MCMC)

To approximate the posterior $P(R|O_x)$ a Monte-Carlo Markov Chain (MCMC) algorithm proposed in [36]. This algorithm generates a sample set of reward function distributions $\{r_1, r_2, \ldots..r_N\}$ according to the targets distribution.

$$P(R|O_x) \approx [1/N] \sum_{i=1}^{N} \delta(R, r_i) \tag{25}$$

These r_i samples are related to a trajectory of a Markov chain. It is considered that its invariant distributions are the counterpart of the target distribution [36]. For a large dimension problem, the drawback of this algorithm is that it requires a large number of sample rewards distributions to guarantee that the estimation is precisely represented by the sample set [37].

4. Conclusion

The reward function is an essential parameter for RL to estimate the best policy. In many applications, it is difficult to specify the true reward function. IRL provide a solution to this problem and has been an attractive field for a researcher for the last decades. It focuses on determining the true underlying reward function for given demonstrations in IE. In this paper, we have presented a conventional model of IRL while some modification and advancement of existing IRL techniques are also discussed here. The experiences of IRL have altered a lot from its first introduction. Improvements empower IRL to be implemented in more complex and practical applications.

List of Acronyms

AI	Artificial Intelligence
MDP	Markov Decision Process
RL	Reinforcement Learning
IRL	Inverse Reinforcement Learning
IL	Imitation Learning
ML	Machine Learning
MLIRL	Maximum Likelihood IRL
MMP	Max-Margin Planning
LP	Linear Programming
IE	Intelligent Environment
LEARCH	LEArning and seaRCH

References

[1] P. Ribino, M. Bonomolo, C. Lodato, and G. Vitale, "A humanoid social robot based approach for indoor environment quality monitoring and well-being improvement," *International Journal of Social Robotics*, pp. 1–20, 2020.

[2] M. Bonomolo, P. Ribino, and G. Vitale, "Explainable post-occupancy evaluation using a humanoid robot," *Applied Sciences*, vol. 10, no. 21, p. 7906, 2020.

[3] C. Lodato and P. Ribino, "A novel vision-enhancing technology for low-vision impairments," *Journal of medical systems*, vol. 42, no. 12, pp. 1–13, 2018.

[4] P. Ribino and C. Lodato, "A norm compliance approach for open and goal-directed intelligent systems," *Complexity*, vol. 2019, 2019.

[5] C. Di Napoli, P. Ribino, and L. Serino, "Customisable assistive plans as dynamic composition of services with normed-qos," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–26, 2021.

[6] P. Ribino and C. Lodato, "A distributed fuzzy system for dangerous events real-time alerting," *Journal of Ambient Intelligence and Humanized Computing*, vol. 10, no. 11, pp. 4263–4282, 2019.

[7] A. Coronato and G. De Pietro, "Tools for the rapid prototyping of provably correct ambient intelligence applications," *IEEE Transactions on Software Engineering*, vol. 38, no. 4, pp. 975–991, 2011.

[8] M. Al-Faris, J. Chiverton, D. Ndzi, and A. I. Ahmed, "A review on computer vision-based methods for human action recognition," *Journal of Imaging*, vol. 6, no. 6, p. 46, 2020.

[9] A. Coronato and G. De Pietro, "Formal design of ambient intelligence applications," *Computer*, vol. 43, no. 12, pp. 60–68, 2010.

[10] A. G. Billard, S. Calinon, and R. Dillmann, "Learning from humans," *Springer handbook of robotics*, pp. 1995–2014, 2016.

[11] A. Coronato, M. Naeem, G. De Pietro, and G. Paragliola, "Reinforcement learning for intelligent healthcare applications: A survey," *Artificial Intelligence in Medicine*, vol. 109, p. 101964, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S093336572031229X

[12] M. Naeem, S. T. H. Rizvi, and A. Coronato, "A gentle introduction to reinforcement learning and its application in different fields," *IEEE Access*, 2020.

[13] L. Pack Kaelbling, M. L. Littman, A. W. Moore, and S. Hall, "Reinforcement Learning: A Survey," *Journal of Artiicial Intelligence Research*, vol. 4, pp. 237–285, 1996.

[14] R. S. Sutton and A. G. Barto, "Reinforcement Learning: An Introduction, Second Edition," 2018, vol. 258, no. 6685, pp. 675–676.

[15] M. Naeem, G. Paragliola, and A. Coronato, "A reinforcement learning and deep learning based intelligent system for the support of impaired patients in home treatment," *Expert Systems with Applications*, p. 114285, 2020.

[16] M. Naeem, G. Paragiola, A. Coronato, and G. De Pietro, "A cnn based monitoring system to minimize medication errors during treatment process at home," in *Proceedings of the 3rd International Conference on Applications of Intelligent Systems*, 2020, pp. 1–5.

[17] G. Paragliola and M. Naeem, "Risk management for nuclear medical department using reinforcement learning algorithms," *Journal of Reliable Intelligent Environments*, vol. 5, no. 2, pp. 105–113, 2019.

[18] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proceedings of the twenty-first international conference on Machine learning*, 2004, p. 1.

[19] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[20] S. Zhifei and E. M. Joo, "A review of inverse reinforcement learning theory and recent advances," in *2012 IEEE congress on evolutionary computation*. IEEE, 2012, pp. 1–8.

[21] A. Coronato and A. Cuzzocrea, "An innovative risk assessment methodology for medical information systems," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–1, 2020.

[22] A. Y. Ng, S. J. Russell *et al.*, "Algorithms for inverse reinforcement learning." in *Icml*, vol. 1, 2000, p. 2.

[23] S. Zhifei and E. M. Joo, "A survey of inverse reinforcement learning techniques," *International Journal of Intelligent Computing and Cybernetics*, 2012.

[24] E. T. Jaynes, "Information theory and statistical mechanics. II," *Physical Review*, vol. 108, no. 2, pp. 171–190, oct 1957.

[25] Z. Shao and M. J. Er, "A review of inverse reinforcement learning theory and recent advances," *2012 IEEE Congress on Evolutionary Computation, CEC 2012*, pp. 10–15, 2012.

[26] B. D. Ziebart, A. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning," in *Proceedings of the National Conference on Artificial Intelligence*, vol. 3, 2008, pp. 1433–1438.

[27] J. Lafferty, A. Mccallum, and F. C. N. Pereira, “Departmental Papers (CIS) Conditional Random Fields : Probabilistic Models for Segmenting and Labeling Sequence Data,” in *Machine Learning*, vol. 2001, no. Icml, 2001, pp. 282–289.

[28] B. D. Ziebart, A. L. Maas, A. K. Dey, and J. A. Bagnell, “Navigate like a cabbie: Probabilistic reasoning from observed context-aware behavior,” in *UbiComp 2008 - Proceedings of the 10th International Conference on Ubiquitous Computing*, 2008, pp. 322–331.

[29] F. S. Melo and M. Lopes, “Learning from demonstration using MDP induced metrics,” in *Machine Learning and Knowledge Discovery in Databases*, vol. 6322 LNAI, no. PART 2, 2010, pp. 385–401.

[30] N. D. Ratliff, D. Silver, and J. A. Bagnell, “Learning to search: Functional gradient techniques for imitation learning,” *Autonomous Robots*, vol. 27, no. 1, pp. 25–53, jul 2009.

[31] D. Silver, J. A. Bagnell, and A. Stentz, “Learning from demonstration for autonomous navigation in complex unstructured terrain,” *International Journal of Robotics Research*, vol. 29, no. 12, pp. 1565–1592, 2010.

[32] C. Pradalier, R. Siegwart, and G. Hirzinger, “Springer Tracts in Advanced Robotics: Preface,” Berlin, Heidelberg, 2011.

[33] J. Oh, Y. Guo, S. Singh, and H. Lee, “Self-imitation learning,” in *International Conference on Machine Learning*. PMLR, 2018, pp. 3878–3887.

[34] H. Ratia, L. Montesano, and R. Martinez-Cantin, “On the Performance of Maximum Likelihood Inverse Reinforcement Learning,” in *arXiv preprint arXiv*, 2012, p. 1202.1558.

[35] A. Boularias, J. Kober, and J. Peters, “Relative entropy Inverse Reinforcement Learning,” in *Journal of Machine Learning Research*, vol. 15, 2011, pp. 182–189.

[36] C. Andrieu, N. De Freitas, A. Doucet, and M. I. Jordan, “An introduction to MCMC for machine learning,” *Machine Learning*, vol. 50, no. 1-2, pp. 5–43, jan 2003.

[37] D. Ramachandran and E. Amir, “Bayesian inverse reinforcement learning,” *IJCAI International Joint Conference on Artificial Intelligence*, pp. 2586–2591, 2007.