INVERSE REINFORCEMENT LEARNING USING EXPECTATION MAXIMIZATION IN MIXTURE MODELS

Jürgen Hahn *Abdelhak M. Zoubir*

Signal Processing Group
Technische Universität Darmstadt
Merckstraße 25, 64283 Darmstadt, Germany
E-mail: {jhahn, zoubir}@spg.tu-darmstadt.de

ABSTRACT

Reinforcement Learning (RL) is an attractive tool for learning optimal controllers in the sense of a given reward function. In conventional RL, usually an expert is required to design the reward function as the efficiency of RL strongly depends on the latter. An alternative has been presented by the concept of Inverse Reinforcement Learning (IRL), where the reward function is estimated from observed data. In this work, we propose a novel approach for IRL based on a generative probabilistic model of RL. We derive an Expectation Maximization algorithm that is able to simultaneously estimate the reward and the optimal policy for finite state and action spaces, which can be easily extended for the infinite cases. By means of two toy examples, we show that the proposed algorithm works well even with a low number of observations and converges after only a few iterations.

Index Terms— Expectation Maximization, Inverse Reinforcement Learning, Markov Decision Process

1. INTRODUCTION

Many applications today lack the flexibility to reproduce or imitate natural behavior. This has become especially important in the recent past for systems that are supposed to adapt to or infer the intent of the user, e.g. in driving assistance applications [1, 2, 3], language understanding [4, 5, 6] or communications [7, 8]. In this work, we present an algorithm that learns from observations to perform complex tasks in the framework of Reinforcement Learning (RL). RL is a powerful tool when only little knowledge about the environment is available. One of the best examples is found in robotics, where agents (robots) are supposed to teach themselves, e.g. how to find optimal paths in a building. However, RL requires knowledge about the reward an agent receives for each state, which can be considered as an attraction to a certain state. In practice, it is often difficult to design the so-called reward function, thus usually expert knowledge is required.

Inverse Reinforcement Learning (IRL), also known as Inverse Optimal Control, aims at solving this problem by learning the reward function from observations. Based on the estimated reward function, the policy, i.e. the optimal action for each state, is derived. One of the first approaches was presented in 2000 by Ng and Sutton [9]. The key idea of their presented algorithm is to alternatively estimate the reward and the policy by iteratively maximizing the difference between the current and previous policy, both depending on the estimated rewards, with respect to derived expected return.

Further methods have been presented in the subsequent years. Similar to the work in [9], Abbeel and Ng presented a max-margin approach and a simplified version, the so-called projection method [10]. A Bayesian approach has been suggested by Ramachandran and Amir [11]. As IRL is an ill-posed problem [9] due to the fact that many reward functions imply the same set of optimal policies, Ramachandran and Amir suggest to average over all possible reward functions where the likelihood is described by an exponential function on the observed trajectories. As prior probability distribution, several application dependent models were investigated. Ziebart *et al.* [12] proposed to solve the ill-posed problem using a maximum entropy based approach. Here, the key idea is to find a distribution of the rewards so that the observed paths are explained best. Levine *et al.* [13] extended this approach to continuous state spaces by employing a Gaussian Process model to interpolate over unobserved states. In contrast to previous work, this approach allows for learning non-linear functions leading to more accurate results.

However, none of the previous methods considered the RL problem in a generative, probabilistic fashion. The original paper by Toussaint and Storky [14] shows how RL can be understood as a Bayesian network, enabling the use of inference techniques for RL [15]. As an example, they propose an efficient Expectation Maximization (EM) [16] algorithm to estimate an optimal policy.

In this work, we show how to use the generative model presented in [14] to simultaneously estimate the reward and policy given observations following an optimal policy. We provide a general problem formulation based on EM and solve it for the case of finite state and action spaces, which leads to

an algorithm that converges after only a few iterations. The proposed framework can directly be used to derive algorithms for the case of infinite state and action spaces.
In Section 2, we provide a brief introduction into Markov Decision Processes as this forms the basis for the mixture model. Section 3 explains how RL can be seen as an inference problem in a Bayesian network. This model is then used in Section 4 to derive an EM algorithm for IRL. The results in Section 5 demonstrate the performance of the algorithm. Finally, a conclusion is drawn and a short outlook is given in Section 6.

2. MARKOV DECISION PROCESSES

Many RL problems can be formulated as Markov Decision Processes (MDPs). A finite MDP is defined by

- a set of N states, $\mathcal{S} = \{s_1, s_2, \ldots, s_N\}$
- a set of M actions, $\mathcal{A} = \{a_1, a_2, \ldots, a_M\}$
- the transition probability $P(s'|s, a)$ describing the probability of entering state s' given state s and taking action a which can be considered as the model of the world and the agent
- the discount factor $\gamma \in [0, 1)$
- and the reward function $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ with absolute value bounded by $R_{\text{max}} \in \mathbb{R}$

In a standard RL problem, the goal is to find the action that maximizes the expected discounted return or value $V(s)$ for each state,

$$V(s) = \mathsf{E}[R(s_{t=0}) + \gamma R(s_{t=1}) + \gamma^2 R(s_{t=2}) + \ldots] \quad (1)$$

i.e. to find the optimal policy. The expected return informs the agent what return to expect when acting according to the policy at time steps $t = 1, 2, \ldots$ and can be estimated by means of the Bellman equations [17, 18]. Note that depending on the application, the reward function depends on the state and the action as well. In contrast to RL, in an IRL setup the reward function $R(s, a)$ is unknown and needs to be estimated.

3. MIXTURE MODEL

Toussaint and Storkey [14] propose to model the joint probability of a trajectory of length K and the obtained rewards, $p(r, s_{1:K}, a_{1:K})$, as a mixture where the mixture components have length $k, k = 1, \ldots, K$ and a reward is obtained only at the end of each component. The probability of a component is given by

$$\begin{aligned} p(r, s_{1:k}, a_{1:k}|k) &= p(s_1)p(a_1|s_1) \\ &\quad \times \prod_{n=2}^{k} p(s_n|s_{n-1}, a_{n-1})p(a_n|s_n). \\ &\quad \times p(r|s_k, a_k) \end{aligned} \quad (2)$$

with known initial state distribution $p(s_1)$.

It can be shown that the discount γ can be represented by a weighting with a geometric distribution $p(k)$ [14], also referred to as time prior. For reasons given in [14], it is convenient to assume that the probability of the reward is equal to the reward itself, i.e. $p(r|s_k, a_k) = R(s, a)$. Hence, it must hold that $R(s, a)$ is bounded by $0 \leq R(s, a) \leq R_{\text{max}} = 1$. The mixture model is then given by the marginal over k, i.e.

$$p(r, s_{1:K}, a_{1:K}) = \sum_{k=1}^{K} p(r, s_{1:k}, a_{1:k}|k)p(k) \quad (3)$$

A key result by Toussaint and Storkey is the fact that the marginal $p(r, s_n, a_n) = \int p(r, s_{1:K}, a_{1:K}) ds_{1:K\setminus n} da_{1:K\setminus n}$ can be efficiently calculated in a recursive manner [14]. Based on this, they proposed an EM algorithm to estimate the parameters θ_π of the policy given as $p(a|s)$.

4. INVERSE REINFORCEMENT LEARNING USING A MIXTURE MODEL

We propose to model the reward function as a mixture with M mixture components and basis functions $\phi_m(s, a)$, $m = 1, \ldots, M$, i.e.

$$R(s, a) = \sum_{m=1}^{M} w_m \phi_m(s, a) = \mathbf{w}^T \mathbf{\Phi}(s, a) \quad (4)$$

with unknown weights $w_m, 0 \leq w_m \leq 1$ and $m = 1, \ldots, M$. The reward r is assumed to be Gaussian distributed with variance σ_r^2, i.e.

$$p(r|s, a) = \mathcal{N}\big(r|R(s, a), \sigma_r^2\big) \ . \quad (5)$$

The goal is to estimate the weights $\mathbf{w}$ of the reward function as well as the parameters θ_π of the policy. The complete set of parameters to be estimated is denoted by $\theta = \{w_1, \ldots, w_M\} \cup \theta_\pi$. Assume we are given a set of L trajectories, each of length K,

$$\mathcal{T} = \{(s_1^{(1)}, \ldots, s_K^{(1)}), \ldots, (s_1^{(L)}, \ldots, s_K^{(L)})\}, \quad (6)$$

that follow an optimal policy, leading the agent to a positive reward. Since the reward and policy are hidden random variables, we propose an EM algorithm similar to [14], while conditioning on the observed trajectories.
Assuming independent trajectories, the complete data likelihood is given by

$$\begin{aligned} \mathfrak{L}(\theta|\mathcal{T}, k, r) &= p_\theta(\mathcal{T}, r, k) \\ &= \prod_{l=1}^{L} p_\theta(a_{1:k}^{(l)}, s_{1:k}^{(l)}, r|k)p(k) \\ &\propto \prod_{l=1}^{L} \left(\prod_{n=1}^{k} p_\theta(a_n^{(l)}|s_n^{(l)}) \right) p_\theta(r|s_k^{(l)}, a_k^{(l)}), \end{aligned}$$

depending on both observed and hidden variables. Thus, the expected log likelihood function Q can be written as

$$\begin{aligned} Q(\theta|\theta') &= \mathsf{E}_{p_{\theta'}(k,r|\mathcal{T})}[\log(p_\theta(\mathcal{T},r,k)] \\ &= \int \sum_{k=1}^{\infty} p_{\theta'}(k,r|\mathcal{T}) \log\left(p_\theta\left(\mathcal{T},r,k\right)\right) dr \\ &= \int \sum_{k=1}^{\infty} p_{\theta'}(k,r|\mathcal{T}) \left(\sum_{l=1}^{L} \Big(\log p_\theta(r|s_k^{(l)},a_k^{(l)}) \right. \\ &\quad \left. + \sum_{n=1}^{k} \log p_\theta(a_n^{(l)}|s_n^{(l)}) \Big) \right) dr + \text{const.} \end{aligned}$$

with θ' denoting the parameters resulting from the previous iteration of the EM algorithm. Recalling the constraints on the weights and reward function, the maximization step is then formulated as a constraint optimization problem,

$$\begin{aligned} &\hat{\theta} = \arg\max_\theta Q(\theta,\theta') \\ &\text{s.t.} \sum_{m=1}^{M} w_m = 1 \\ &\quad w_m \geq 0 \quad \forall m \in \{1,\dots,M\}. \end{aligned} \tag{7}$$

In order to solve the sum constraint, a Lagrangian function is used, i.e.

$$\mathcal{L}(\mathbf{w},\lambda) = Q(\theta,\theta') + \lambda(1 - \sum_{m=1}^{M} w_m) \tag{8}$$

with Lagrangian multiplier λ. Calculating $\frac{\partial \mathcal{L}}{\partial w_i}$ for $i = 1,\dots,P$, we obtain

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial w_i} &= \int \sum_{k=1}^{\infty} p_{\theta'}(k,r|\mathcal{T}) \left(\sum_{l=1}^{L} -\frac{1}{\sigma_r^2} \left(r - \mathbf{w}^T \mathbf{\Phi}(s_k^{(l)},a_k^{(l)}) \right) \right. \\ &\quad \left. \left(-\Phi_i(s_k^{(l)},a_k^{(l)}) \right) dr \right) - \lambda. \\ &= -\frac{1}{\sigma_r^2} \sum_{k=1}^{K} p(k) \left(\prod_{l=1}^{L} R_{\theta'}(s_k^{(l)},a_k^{(l)}) \right) \sum_{l=1}^{L} \Phi_i(s_k^{(l)},a_k^{(l)}) \\ &\quad + \frac{1}{\sigma_r^2} \sum_{k=1}^{\infty} p(k) \sum_{l=1}^{L} \mathbf{w}^T \mathbf{\Phi}(s_k^{(l)},a_k^{(l)}) \Phi_i(s_k^{(l)},a_k^{(l)}) - \lambda \end{aligned}$$

where we assumed that k and r are independent and k is independent of $\mathcal{T}$. Further, recall that the expected value of r is $R(s,a)$. Replacing $\lambda = \frac{1}{\sigma_r^2}\tilde{\lambda}$ and $\prod_{l=1}^{L} R_{\theta'}(s_k^{(l)},a_k^{(l)}) = R_{\theta'}^{\mathcal{T}}$ and setting $\frac{\partial \mathcal{L}}{\partial w_i} = 0$, results in

$$\begin{aligned} \tilde{\lambda} = \sum_{k=1}^{\infty} p(k) \left(R_{\theta'}^{\mathcal{T}} \sum_{l=1}^{L} \Phi_i(s_k^{(l)},a_k^{(l)}) \right. \\ \left. - \sum_{l=1}^{L} (\mathbf{w}^T \mathbf{\Phi}(s_k^{(l)},a_k^{(l)}) \Phi_i(s_k^{(l)},a_k^{(l)}) \right). \end{aligned} \tag{9}$$

The Lagrangian multiplier $\tilde{\lambda}$ can be substituted by any of the P equations, e.g. the first. Then, we obtain for the remaining equations, $i = 2,\dots,P$,

$$\begin{aligned} &\sum_{k=1}^{\infty} p(k) \left(R_{\theta'}^{\mathcal{T}} \sum_{l=1}^{L} \Phi_i(s_k^{(l)},a_k^{(l)}) - \Phi_1(s_k^{(l)},a_k^{(l)}) \right) \\ &\quad = \mathbf{w}^T \sum_{k=1}^{\infty} p(k) \left(\sum_{l=1}^{L} (\mathbf{\Phi}(s_k^{(l)},a_k^{(l)}) (\Phi_i(s_k^{(l)},a_k^{(l)}) - \Phi_1(s_k^{(l)},a_k^{(l)}) \right) \end{aligned}$$

and with $\frac{\partial L(\mathbf{w},\lambda)}{\partial \lambda} = 1 - \sum_{m=1}^{M} w_m$, this can be written as an equation system,

$$\begin{aligned} &\mathbf{A}\mathbf{w} = \mathbf{b} \\ &\text{s.t. } w_m \geq 0 \quad \forall m \in \{1,\dots,M\} \end{aligned} \tag{10}$$

where

$$\mathbf{A} = \begin{bmatrix} \sum_k^{\infty} p(k) \left(\sum_l^L (\mathbf{\Phi}^T(s_k^{(l)},a_k^{(l)}) (\Phi_2(s_k^{(l)},a_k^{(l)}) - \Phi_1(s_k^{(l)},a_k^{(l)}) \right) \\ \vdots \\ \sum_k^{\infty} p(k) \left(\sum_l^L (\mathbf{\Phi}^T(s_k^{(l)},a_k^{(l)}) (\Phi_M(s_k^{(l)},a_k^{(l)}) - \Phi_1(s_k^{(l)},a_k^{(l)}) \right) \\ [1,\dots,1] \end{bmatrix}^T$$

and

$$\mathbf{b} = \begin{pmatrix} \sum_k^{\infty} p(k) \left(R_{\theta'}^{\mathcal{T}} \sum_l^L \Phi_2(s_k^{(l)},a_k^{(l)}) - \Phi_1(s_k^{(l)},a_k^{(l)}) \right) \\ \vdots \\ \sum_k^{\infty} p(k) \left(R_{\theta'}^{\mathcal{T}} \sum_l^L \Phi_M(s_k^{(l)},a_k^{(l)}) - \Phi_1(s_k^{(l)},a_k^{(l)}) \right) \\ 1 \end{pmatrix}.$$

Eq. (10) can be efficiently solved by a non-negative least squares algorithm [19]. Since an estimate of the reward is given now, the algorithms presented in [14] or [15] can be used to estimate the parameters, θ_π, of the policy.

Note that, in practical problems the product in $R_{\theta'}^{\mathcal{T}}$ is likely to tend to zero and may cause numerical problems. Since we assume that the observed trajectories lead to a high positive reward, the product can be approximated by $\prod_{l=1}^{L} R(s_k^{(l)},a_k^{(l)}) = (R_{\max})^L = 1$. Thus, the estimate of the weights does no longer depend on previous iterations. Then, an initialization of the weights is not required since an estimate of weights is given as a closed-form solution.

5. SIMULATIONS

In order to demonstrate the performance of the presented framework and the derived algorithm, we show two toy examples similar as the ones in [10]. Since we consider finite state and action spaces, we choose a simple dirac function as the basis function for the reward.

First, we consider the maze shown in Fig. 1 with rewards in the upper right, upper left, lower right corners and the

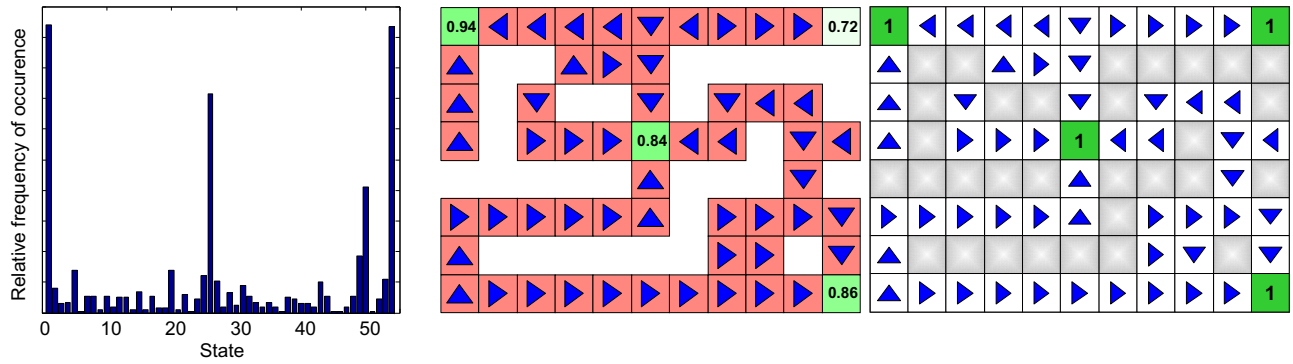

Fig. 1: Histogram of observed states (left), estimated rewards (center) for $L = 100$, policy and true reward of the maze example (right). Arrows indicate the policy and zero reward.

center (highlighted in green) and a state space consisting of 57 elements. We allow the agent to take the actions of going north, south, east and west where there is a 30% chance of failure, meaning that the agent moves in a random direction. Probabilities that would lead outside the maze are added to the probability of staying in the current state to ensure that the agent is always in a valid state.

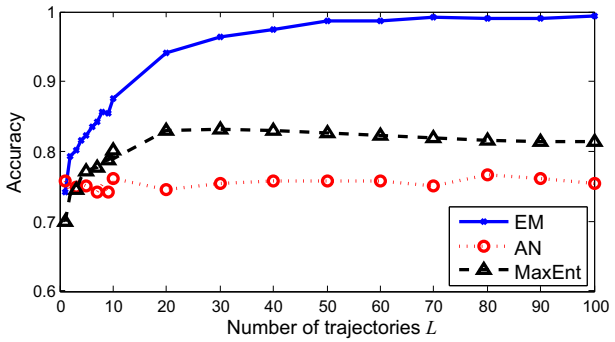

Fig. 2: Accuracy of reproducing the policy for the maze

Applying the proposed algorithm for $L = 100$ observations, we obtain the estimated reward shown in Fig. 2. Inspecting the histograms of the given trajectories, we observe that only three states have been visited frequently. The result shows that the reward function has been well reconstructed. Further, high accuracy in reproducing the policy from observed data is achieved even for low number of observations, outperforming other algorithms (IRL-EM:proposed, AN:[10], MaxEnt:[12]).

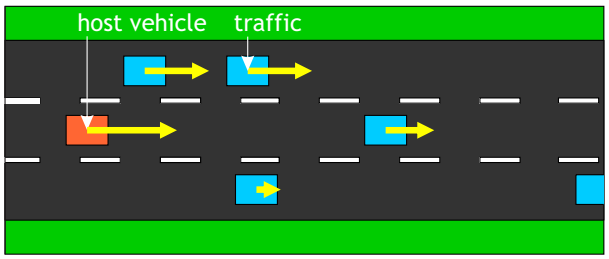

Fig. 3: Screenshot of the driving simulator

Second, we implemented a simple highway scenario as in [10]. The agent, the host vehicle, is driving on a highway with three lanes (Fig. 3). The goal is to drive as fast as possible without causing an accident. The speed of the traffic depends on the lane the vehicles travel on, increasing from the right to left lane. A dynamic scenario is given by random lane changes of the vehicles. Note that the host vehicle aims at driving faster than the traffic. Since accidents shall be avoided and overtaking maneuvers are not always possible, the host can take four different speeds. However, if the lanes are free, the host may change the lane as well (and at the same time change the speed), resulting in twelve different actions in total. A state is defined by the current lane and velocity of the host vehicle, and the information whether the lanes are blocked, leading to 36 states in total. We used 200 snapshots with a length of 5 steps from three different expert demonstrations to learn the following driving styles:

- aggressive: as fast as possible, overtaking on all lanes but avoid collisions
- bad: hit as many obstacles as possible
- legal: avoid collision and do not overtake from the right

The true rewards for these driving styles are unknown, thus a numerical evaluation is difficult. However, with a sufficient number of training data (basically all states have to be covered), the agent is well able to reproduce the demonstrated driving style. Short videos for the different driving styles can be found on http://www.spg.tu-darmstadt.de/res/dl/.

6. CONCLUSION

Based on a mixture model framework for Reinforcement Learning, we presented an algorithm to estimate the reward and policy given observations. Thus, rewards as in classical RL do not need to be defined by an expert in advance, but are estimated from observed optimal behavior. The presented algorithm is based on a probabilistic generative model, enabling the use of a broad range of inference techniques for IRL. The results show that the algorithm works well in finite state spaces. Future research will concentrate on transferring these results to infinite state spaces and easing the limitations on the reward function, increasing the applicability of this framework to a wider field of problems.

7. REFERENCES

[1] M. Liebner, F. Klanner, M. Baumann, C. Ruhhammer, and C. Stiller, "Velocity-based driver intent inference at urban intersections in the presence of preceding vehicles," *IEEE Intelligent Transportation Systems Magazine*, vol. 5, no. 2, pp. 10–21, Summer 2013.

[2] F. Muehlfeld, I Doric, R. Ertlmeier, and T. Brandmeier, "Statistical behavior modeling for driver-adaptive pre-crash systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 4, pp. 1764–1772, Dec 2013.

[3] J. Wang, L. Zhang, D. Zhang, and K. Li, "An adaptive longitudinal driving assistance system based on driver characteristics," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 1, pp. 1–12, March 2013.

[4] C. Lin and M. Kan, "Adaptive fuzzy command acquisition with reinforcement learning," *IEEE Transactions on Fuzzy Systems*, vol. 6, no. 1, pp. 102–121, Feb 1998.

[5] Y. Pan, H. Lee, and L. Lee, "Interactive spoken document retrieval with suggested key terms ranked by a markov decision process," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 2, pp. 632–645, Feb 2012.

[6] P. Su, Y. Wang, T. Yu, and L. Lee, "A dialogue game framework with personalized training using reinforcement learning for computer-assisted language learning," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2013, pp. 8213–8217.

[7] J. Lunden, S.R. Kulkarni, V. Koivunen, and H.V. Poor, "Multiagent reinforcement learning based spectrum sensing policies for cognitive radio networks," *IEEE Journal of Selected Topics in Signal Processing*, vol. 7, no. 5, pp. 858–868, Oct 2013.

[8] N. Mastronarde and M. van der Schaar, "Fast reinforcement learning for energy-efficient wireless communication," *IEEE Transactions on Signal Processing*, vol. 59, no. 12, pp. 6262–6266, Dec 2011.

[9] A. Y. Ng and S. J. Russell, "Algorithms for inverse reinforcement learning," in *Proceedings of the Seventeenth International Conference on Machine Learning*, San Francisco, CA, USA, 2000, ICML '00, pp. 663–670, Morgan Kaufmann Publishers Inc.

[10] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proceedings of the Twenty-first International Conference on Machine Learning*, New York, NY, USA, 2004, ICML '04, ACM.

[11] D. Ramachandran and E. Amir, "Bayesian inverse reinforcement learning," in *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, San Francisco, CA, USA, 2007, IJCAI'07, pp. 2586–2591, Morgan Kaufmann Publishers Inc.

[12] B. D. Ziebart, A. Maas, J. . Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning," in *Proceedings of the 23rd National Conference on Artificial Intelligence*. 2008, vol. 3 of *AAAI'08*, pp. 1433–1438, AAAI Press.

[13] S. Levine, Z. Popovic, and V. Koltun, "Nonlinear inverse reinforcement learning with gaussian processes," in *Advances in Neural Information Processing Systems*, 2011, pp. 19–27.

[14] M. Toussaint and A. Storkey, "Probabilistic inference for solving discrete and continuous state markov decision processes," in *Proceedings of the 23rd International Conference on Machine Learning*, New York, NY, USA, 2006, ICML '06, pp. 945–952, ACM.

[15] M. W. Hoffman, *Decision making with inference and learning methods*, Ph.D. thesis, University Of British Columbia, 2013.

[16] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society, Series B*, vol. 39, no. 1, pp. 1–38, 1977.

[17] R. Bellman, *Dynamic Programming*, Princeton University Press, Princeton, NJ, USA, 1 edition, 1957.

[18] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*, MIT Press, Cambridge, MA, USA, 1998.

[19] C. L. Lawson and R. J. Hanson, *Solving Least Squares Problems*, Series in Automatic Computation. Prentice-Hall, Englewood Cliffs, NJ 07632, USA, 1974.