MULTITASK IMITATION LEARNING: A CLOSER LOOK

BY

RUSHIT N. SHAH

Department of Computer Science
University of Illinois at Chicago
Chicago, IL 60607

WRITTEN CRITIQUE

Abstract

Techniques which enable agents to learn from expert demonstrations, also known as imitation learning, have been shown to perform successfully. However, for generalist agents, which must be able to learn to perform multiple different tasks, it may be tedious to learn multiple reward functions for each task separately. In contrast to traditional imitation learning, multitask imitation learning allows an agent to learn multiple reward functions simultaneously, to quickly adapt to a variety of tasks, rather than just one. This study takes a closer look at three multitask imitation learning techniques developed over the last decade. These techniques differ significantly from each other in how they address multitask imitation. Additionally, the development of the techniques under review in this study also spans the last decade; this serves to provide a view of this research area as it evolved The incremental changes in approach are also highlighted as these techniques are presented. First an overview of imitation learning is provided. The three techniques considered are then detailed and critiqued, with emphasis on the differences in the approaches and major shortcomings of each. Finally, new research avenues which may lend themselves to the application of multitask imitation learning, are briefly discussed.

Table of Contents

Chapter 1 Introduction **1**

Chapter 2 Notation, Background and Definitions **3**
2.1 Markov Property 3
2.2 Markov Decision Process 3
2.3 Imitation Learning Formulation 4

Chapter 3 Related Work **5**

Chapter 4 Apprenticeship Learning About Multiple Intentions [1] **7**
4.1 Maximum Likelihood Inverse Reinforcement Learning (MLIRL) 7
4.2 Multitask Learning (EMIRL) 8
4.3 Contributions and Critique 10

Chapter 5 One-Shot Visual Imitation via Meta-Learning [2] **11**
5.1 Model-Agnostic Meta-Learning (MAML) 11
5.2 Meta-Imitation Learning Using MAML 12
5.3 Contributions and Critique 12

Chapter 6 Lifelong Inverse Reinforcement Learning [3] **14**
6.1 Maximum Entropy Inverse Reinforcement Learning (MaxEnt IRL) 14
6.2 Lifelong IRL 15
6.3 Contributions and Critique 17

Chapter 7 Future Work **18**

Chapter 8 Conclusion **19**

References **20**

Chapter 1

Introduction

Reinforcement learning (RL) is an important area of machine learning research, where an agent interacts with its environment by following a policy. In each state of the environment it takes an action based on the policy, and as a result, receives a reward and transitions to a new state. The goal of RL is to learn an optimal policy which maximizes the long-term cumulative rewards. RL algorithms used the received reward signal to approximate the best policy for the agent . While these methods generally perform well with impressive empirical results in domains [4, 5], there are several scenarios under which these algorithms fail to retrieve at the optimal policy. For instance, in environments where rewards are sparse (for example when reward is only received upon reaching a terminal state), an RL agent does not receive adequate feedback to learn which actions are the right ones to take. RL approaches are also data inefficient [**?**], often requiring extensive interaction with complex environments via trying suboptimal actions before the agent can learn the negative consequences of those actions. More generally, RL approaches assume that a reward function for the target task is available, but the manual design of which may be extremely challenging in cases where the environment and the action space is complex.

An alternative to RL is imitation learning (IL). While manually designing reward functions in order to obtain autonomous behavior might be intractable, the demonstration of the expected behavior is not. The purpose of imitation learning is to efficiently learn a desired behavior by imitating an expert's behavior from the expert's demonstrations [6, 7, 8, 9, 10, 11].

An important assumption underlying most imitation algorithms is that the demonstrations which the agent is trying to imitate are generated by an expert who is assumed to be acting optimally with respect to some reward function which is unknown to the agent. One way to recover the policy which reproduces the demonstrated behavior is to learn a function which maps the inputs (states) directly to an action/trajectory. This reduces the problem of imitation learning to one of supervised learning, and is often referred to as Behavior Cloning (BC) [12]. Alternatively, the agent may use the demonstrations to recover the expert's unknown reward function; this problem is known as Inverse Reinforcement Learning (IRL) [13] or Inverse Optimal Control (IOC) [14].

BC and IRL approaches are designed for agents which must learn a single task. However, a more versatile agent must learn many tasks via demonstrations. While it is possible for such an agent to learn these tasks in isolation (by adopting a BC or IRL strategy for each task separately), this process may be tedious, especially given the large number of demonstrations required to learn even a single task via imitation learning. For instance, consider a personal domestic service robot tasked with performing with household chores. Such an agent must learn a variety of tasks such as sweeping the floor, loading the dishwater, and setting up the dining table. Providing demonstrations to learn these tasks in isolation may be cumbersome for the human [3]. Besides, there might be useful information derived from learning one task which might be applicable to other future tasks. Such sharing might get inhibited if the tasks are learned separately. For example, any action which leads to dropping a fragile kitchenware must be negatively rewarded and consequently avoided, whether it be while the agent is learning to load a dishwasher, or setting-up a dining table. Leveraging this information can not only help reduce redundancy, but might also reduce the number of demonstrations needed to learn future tasks. These challenges have led to the development of *Multitask Imitation Learning*.

This study reviews multitask imitation learning techniques. While several techniques are discussed, three specific techniques are reviewed in greater detail; these techniques vary vastly in their approach to the multitask imitation learning problem thereby covering the breadth of approaches; they are also representative of the chronological development of this research area over time. The rest of this study is divided as follows—relevant background, definitions and notation as they pertain to imitation learning are provided in chapter 2. Related work is discussed in Chapter 3. Chapters 4, 5, and 6 are dedicated to the detailed review of the three chosen multitask imitation learning approaches; here the methods are described in detailed, compared, and are critiqued. Chapter 7 provides a discussion and the scope for future work. Chapter 8 concludes.

Chapter 2

Notation, Background and Definitions

Notation and mathematical formulation of the imitation learning problem is first described in this Chapter; this includes the description of the Markov property and the Markov Decision Process which is central to the imitation learning problem definition. The imitation learning problem is then mathematically formulated.

2.1 Markov Property

A sequence of system states $s_0, \dots, s_t$ is called a Markov chain if at any given time t, the conditional probability distribution of the state s_t depends only on state s_{t-1}, and is independent of all the states in the sequence preceding s_{t-1} [15].

2.2 Markov Decision Process

A Markov decision process (MDP) is defined as a tuple $\langle \mathcal{S}, \mathcal{A}, T, r, \gamma \rangle$, where $\mathcal{S}$ is a set of states, $\mathcal{A}$ is a set of actions, T defines the transition function/system dynamics defined as $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0, 1]$ gives the probability $P(s_{t+1}|s_t, a_t)$ of reaching state s_{t+1} when taking action a_t in state s_t at time t. The reward function is defined as $r : \mathcal{S} \mapsto \mathbb{R}$. The policy $\pi : \mathcal{S} \times \mathcal{R} \mapsto [0, 1]$ denotes the probability distribution $P(a_t|s_t)$ which quantifies the probability of choosing action a_t in state s_t; it models the probability distribution over all the possible actions in a given state. $\gamma \in [0, 1)$ is the discount factor which weights the rewards received by the agent in the states visited in the future. If all the components of an MDP are well-defined, it can be solved to obtain an optimal policy π^*. This policy π^* is chosen such that it maximizes the expected sum of discounted rewards earned by an agent, by following that policy, starting from an initial state s_0,

$$\pi^* = \arg\max_{\pi} V^{\pi}(s_t) = \arg\max_{\pi} \mathbb{E}_{\pi}\Big[\sum_t \gamma^t r(s_t)\Big]$$

Where V^{π} is referred to as the value function [16]. A policy may also be obtained by maximizing the action-value function defined as $Q^{\pi}\ (s_t, a_t)$, which denotes the expected sum of rewards starting from an initial state s_t, and taking action a_t in that state. These methods form the basis of reinforcement learning [16, 17, 18].

2.3 Imitation Learning Formulation

Unlike the reinforcement learning setting, the reward function r of the MDP is unknown in the imitation learning setting. The MDP in this case is defined as $MDP \setminus r = \langle \mathcal{S}, \mathcal{A}, T, \gamma \rangle$. Information regarding this unknown reward function must be inferred from the set of demonstrations $\mathcal{D} = \{\tau_1, \tau_2, \ldots, \tau_n\}$ provided by an expert, where each demonstration τ_i is provided as a sequence of state-action pairs $\{(s_0, a_0), \ldots, (s_t, a_t)\}$. Each demonstration is generated by the expert's unknown policy π^*, which is (usually) assumed to be optimal.

In the imitation learning setting known as Behavioral Cloning (BC) [12], the expert's set of demonstrations $\mathcal{D}$ is used to learn a direct mapping $\hat{\pi}$ from states to actions as

$$a_t = \hat{\pi}(s_t)$$

This approach closely resembles the traditional supervised learning approach where the mapping learned is from features to labels, given a data set. In the Inverse Reinforcement Learning (IRL) [13] setting, the expert's unknown reward function r must first be estimated using the given set of expert demonstrations $\mathcal{D}$. Once the reward function is known, the MDP can be considered complete and can be solved for an optimal policy using traditional reinforcement learning approaches. BC and IRL comprise the two broad classes of imitation learning methods, and the choice between the two comes down to the problem setting and the representation of the expert demonstrations. Further details about the differences between the two approaches and when to choose one over the other are provided in [19]. Further, in the IRL setting, the reward function r is typically parameterized by a vector of reward weights θ, which can then be applied to a feature vector $\phi(s_t)$ describing a given state. Thus, the reward function may then be written as $r_\theta(s_t) = \theta^T \phi(s_t)$.

Chapter 3

Related Work

The three techniques discussed in this study cover both BC and IRL. Hence, in this Chapter, popular BC and IRL techniques are introduced first. This is followed by a review of the work done on multitask imitation learning, apart from the three techniques discussed in this study.

As stated before, given expert demonstration trajectories τ_i consisting of a set of encountered states, and expert actions, the problem of behavioral cloning [12, 20, 21] is reduced to that of supervised learning, where a mapping from the states to actions is learned using either a classifier or a regressor [22]. Examples of success stories of BC include [23, 24] where the input states were provided as images, owing to which the BC policies in these studies were modeled as deep convolutional neural networks. However, as the name suggests, behavioral cloning trains a policy that learns to imitate an expert, and tends to fail in scenarios where a state is encountered which the policy was not trained on. This well-known problem with BC and is referred to as covariate shift [22].

IRL techniques seek to learn the reward function which the expert is assumed to be trying to maximize while generating the demonstrations i.e. the assumption is that the policy of the expert is (nearly) optimal under this unknown reward function. However, recovering the expert policy is an ill-posed problem since there exist infinitely many reward functions under which the expert policy is optimal. Developed approaches thus make additional assumptions in order to find solutions which generalize well. A class of methods known as maximum-margin [25] methods seek to maximize the sum of differences between the value of the optimal action and that of the next-best action for every state. Maximum margin planning (MMP) [26] defines its loss function as the closeness between the learned and the demonstrated behavior, in terms of the state visitation frequency. Maximum entropy IRL [27, 28, 29] deals with the IRL ambiguity in a probabilistic way by assuming that the expert behaves according to a policy which maximizes entropy. This approach parameterizes this distribution over demonstrations and then optimizes for parameters which maximize the likelihood of observing the given expert demonstrations under that distribution. The distribution thus obtained has two advantages; first, it does not differentiate between trajectories which yield the same reward, and second, it exponentially prefers demonstrations which yield higher rewards. While maximum entropy

IRL generates a distribution over trajectories, Bayesian IRL [30] make the assumption that the demonstrator is following a softmax policy, and estimates posterior probabilities over possible reward functions. Within the active learning framework, the demonstrations are provided dynamically; and the learner queries the expert for additional demonstrations when needed [31].

However, the BC and IRL approaches discussed so far are aimed at learning a single reward function. While agents trained using these techniques learn to perform well on the trained task, they are not versatile when it comes to learning to perform other tasks. Within the frameworks and methods described thus far training an agent to perform multiple tasks would typically involve the agent learning multiple reward functions, one corresponding to each task, in isolation. Since this endeavor would require large amounts of expert observations for each task separately, it does not lend itself to practical implementation. Indeed, versatile, generalist robots, which can be trained to perform multiple tasks, or can learn how to perform new tasks using minimal training and demonstrations, are the need of the future. In this context, multitask IRL methods have been developed which allow agents to learn multiple tasks at the same time using demonstrations of those tasks, or at the very least are versatile enough to allow the agents to learn new tasks with minimal new demonstrations. While this study discusses three approaches that do exactly this, this Chapter highlights some other efforts made to achieve this. One of the earliest efforts to tackle multitask IRL leveraged the Bayesian IRL setting [32] in which the authors defined a joint prior on rewards and policies, and learned multiple tasks by sampling from that prior. One approach to multitask IRL is to assume that the demonstrations for each task are generated via hidden intentions; to this end unsupervised learning was used to cluster demonstrations across multiple tasks using Dirichlet process mixture modeling, with each cluster representing an intention. When a new demonstration was obtained, it was assigned into one of the discovered intentions/clusters of demonstrations, to learn a policy for the new task quickly [33].

In the next three chapters, we take a closer look at three distinct techniques developed for multitask imitation learning. These techniques vary in several ways. First, they are significantly different in their approach to solving the multitask imitation problem; the first employs an unsupervised approach to learning new tasks, the second uses a one-shot approach which allows new tasks to be learnt using a single new demonstration; the third approach aims to learn a representation of multiple reward functions in a way that they are all learnt together, and information from previously-learned tasks is transferable to new tasks. The variety in these approaches provides insight into the breadth of multitask imitation approaches developed over the past decade. Second, since the three techniques were developed over a decade, summarizing and critiquing them helps characterize the evolution of this research area—to this end these techniques are presented here in chronological order of their development.

Chapter 4

Apprenticeship Learning About Multiple Intentions [1]

Multitask imitation aims to learn the unknown reward functions underlying the expert-generated demonstrations, each of which corresponds to a different task being executed. One of the earlier attempts at multitask imitation learning, the work in [1] focuses on doing exactly this. The developed approach is quite general, and is applicable to all multitask imitation scenarios, from robotic grasping, to autonomous driving. The novelty of this method lies in the fact that the demonstrations used are not labeled i.e. it is not known a priori which of the candidate tasks a particular demonstration corresponds to. This does reflect a real-world scenario, where it is, in fact, expensive to label each demonstration in the training data. In the light of this challenge, the authors adopt an unsupervised learning approach which is detailed next.

4.1 Maximum Likelihood Inverse Reinforcement Learning (MLIRL)

Given an $MDP \setminus r = \langle \mathcal{S}, \mathcal{A}, T, \gamma \rangle$, set of expert demonstrations $\mathcal{D} = \{\tau_1, \tau_2, \ldots, \tau_n\}$, and a reward function parameterized by θ as $r_\theta(s) = \boldsymbol{\theta}^T \phi(s_t)$, the authors in (Babes et al. 2011) first introduce the Maximum Likelihood Inverse Reinforcement Learning (MLIRL) to learn the learner's reward parameters θ_L which maximize the likelihood of observing the expert's demonstrations. Specifically, the Q-value of a state-action pair is modified to facilitate Boltzmann exploration [34] as

$$Q_{\theta_L}(s,a) = \theta_L^T \phi(s,a) + \gamma \sum_{s'} T(s,a,s') \sum_{a'} \frac{Q(s,a') e^{\beta Q(s,a')}}{\sum_{a''} e^{\beta Q(s,a'')}}$$

Boltzmann exploration makes the agent tend to choose action which it deems more rewarding, by assigning an exponentially higher probability to such actions. The log likelihood of the trajectories under the Boltzmann policy $\pi_{\theta_L}(s,a) = e^{\beta Q(s,a)} / \sum_{a'} e^{\beta Q(s,a')}$ is given as

$$L(D|\theta) = \log \prod_{i=1}^{n} \prod_{(s,a)\in\tau_i} \pi_\theta(s,a)^{w_i} = \sum_{i-1}^{n} \sum_{(s,a)\in\tau_i} w_i \log \pi_\theta(s,a)$$

Algorithm 1 Maximum Likelihood IRL

```
    Input MDP\r, features $\phi$, demonstrations $\mathcal{D} = \{\tau_1, \tau_2, \dots, \tau_n\}$, weights $\{w_1, w_2, \dots, w_n\}$
    Output Return $\theta_L = \theta_M$
    Initialize Choose a random set of reward weight $\theta_1$
1: procedure
2:     for $t = 1$ to $M$ do
3:         Compute $Q_{\theta_t}, \pi_{\theta_t}$
4:         Compute $L = \sum_i w_i \sum_{(s,a)\in\tau} \log[\pi_{\theta_t}(s,a)]$
5:         Update $\theta_{t+1} \leftarrow \theta_t + \alpha_t \nabla L$
```

where w_i is the weight encoding of the demonstration frequency of demonstration i. The final reward parameters θ chosen are the ones which maximize the aforementioned likelihood of observing the expert's demonstrations in $\mathcal{D}$. The MLIRL method is summarized in Algorithm 1. The parameters are updated via gradient ascent, which is possible due to the Boltzmann exploration modification of the Q-value formulation. The MLIRL method is found in some ways to be similar to the maximum entropy IRL [27], in that both techniques attempt to maximize the likelihood of observing the expert demonstrations, but under different loss functions. The authors in [1] claim that the behavior of the expert is learned by the agent due to the fact that the optimization in Algorithm 1 learns those weights which assign high probability to observed expert behavior, and lower probabilities to unobserved behavior.

4.2 Multitask Learning (EMIRL)

In [1], the authors make the assumption that even though a set of expert demonstrations corresponding to different tasks is available, it is not known which task was being executed when the expert generated any particular demonstration i.e. the demonstrations do not contain task labels. This notion is formalized as follows - it assumed that there exists a finite set of K tasks, the reward function of each of which is parameterized by vector θ_k. Further, it is also assumed that the set of n expert demonstrations contains at least one demonstration of each of the K tasks ($N > K$).

The authors then adopt a clustering approach to the problem of missing task labels. The proposed method hypothesizes that demonstrations performed when executing a particular task, when clustered together, can be used to learn the reward function for that task. The authors use soft-clustering via the popular expectation-maximization (EM) algorithm [35]. The problem is setup as follows - Define z_{ij} be the probability that demonstration i is generated when executing task j. Let θ_j be the estimate of the reward function parameters for the task (also a cluster) j, and let ρ_j be the prior probability of cluster j. A parameter vector being sought is defined as $\Theta = (\rho_1, \dots, \rho_K, \theta_1, \dots, \theta_K)$, and Θ^t is the parameter vector at time t. Additionally, $y_i = j$ if a particular demonstration i was generated while executing task j, and $y = (y_1, y_2, .., y_n)$. Finally

Algorithm 2 EM Trajectory Clustering (EMIRL)

```
    Input Input: Demonstrations $\mathcal{D} = \tau_1, \tau_2, \ldots, \tau_n$, number of tasks $K$
    Initialize Randomly initialize $\rho_1, ..., \rho_K, \theta_1, ..., \theta_K$
1: procedure
2:     while target number of iterations is not completed do
3:         E step: Compute $z_{ij}^t = \prod_{(s,a)\in\tau_i} \pi_{\theta_j^t}(s,a)\rho_j^t/z$
4:         M step: For all $l$, $\rho_l = \sum_i z_{il}^t$.
5:         Update $\theta_t$ via MLIRL on $D$ with weight $z_{il}$ on demonstration $i$.
```

$z_{ij}^t = P(\tau_i|\theta_j^t)$ is defined as the probability of demonstration i having been generated while executing task j. z_{ij}^t is the measure of belongingness or membership of demonstration i to cluster/task j. In the E-step of the EM method the expected values of this degree of membership as

$$z_{ij}^t = \prod_{(s,a)\in\tau_i} \frac{\pi_{\theta_j^t}(s,a)\rho_j^t}{Z}$$

Where Z is the normalization factor. The M-step then involves choosing a value of Θ which maximizes the EM Q-function which the authors define as

$$\begin{aligned} Q(\Theta, \Theta^t) &= \sum_y L(\Theta|D,y)P(y|D,\Theta^t) \\ &= \sum_{l=1}^{K}\sum_{i=1}^{N} \log{(\rho_1)} z_{il}^t + \sum_{l=1}^{K}\sum_{i=1}^{N} \log{[P(\tau_i|\theta_l)]} z_{il}^t \end{aligned}$$

Since the two terms in the EM Q-function are not dependent, the authors maximize them independently to obtain the following solutions

$$\rho_l^{t+1} = \frac{\sum_i z_{il}^t}{n}$$

$$\theta_l^{t+1} = \arg\max_\theta \sum_i z_{il}^t \log{[P(\tau_i|\theta_l)]}$$

The second update equation can be written as

$$\begin{aligned} \theta_l^{t+1} &= \arg\max_\theta \sum_i z_{il}^t \log{[P(\tau_i|\theta_l)]} \\ &= \arg\max_\theta \sum_i z_{il}^t \log{[\pi_{\theta_1}(s,a)]} \end{aligned}$$

Which is the same as the likelihood function being maximized in MLIRL (see Algorithm 1). The degree of membership $z_i l^t$ here corresponds to the trajectory specific weight-encoding w_i in MLIRL. The training process for the multitask IRL using MLIRL is summarized in Algorithm 2.

4.3 Contributions and Critique

The main contribution of the work in [1] is the unsupervised clustering of the expert's demonstrations. Indeed, the developed method does not require the labeled trajectories to be labeled as belonging to a known task, and the method clusters similar trajectories into groups, and the demonstrations in a particular group are assumed to belong to the same task, and are used to learn the expert's unknown reward function and a policy for that task. However, the authors do not describe how the policy parameters for a particular task must be updated when new demonstrations are added to the clusters; if the assumption is that the policy parameters learned during training are near-optimal and do not require further updating, then this might require a large number of demonstrations per cluster, the existence of which cannot be known or guaranteed a priori.

A major drawback of the developed method is that given that it involves a traditional EM clustering of demonstrations, the number of tasks K represented in the demonstration data set must be known a priori; the method is then restricted to learning reward parameters and policies for only this fixed set of K tasks. Indeed, if an expert demonstration is encountered that does not represent any of the K tasks well, such a demonstration might be detrimental to the policy parameters of all K tasks. Additionally, the developed method also assumes that all the K tasks share a common state-action space. This restricts the applicability of the developed method to very similar tasks, and does not allow learning tasks which may be similar in nature to an expert, but have demonstrations which differ in their representation.

Maximum likelihood methods for imitation learning were developed before the authors in [1] used it in their work. The same applies to the modeling of the probability distribution of the demonstrations as an exponential distribution. In fact, these contributions date as far back as Bayesian IRL [30] and maximum entropy IRL [27]. The application of these techniques to multitask imitation learning is novel in [1]. However, even the application of these concepts in the developed method is not novel; the use of maximum likelihood and exponential family of distributions is in the same context as it is in Bayesian IRL and maximum entropy IRL.

Chapter 5

One-Shot Visual Imitation via Meta-Learning [2]

The imitation learning approached detailed in the previous chapter enabled an agent to learned multiple tasks during the learning procedure. The approach resulted in the expert demonstrations being clustered into groups, each representing a particular task—extending the agent's knowledge this required a complete retraining using demonstrations of old tasks along with those for new tasks. The method developed in [2] deviates from that approach, in that it primes the agent to quickly learn new behaviors. In other words, at the end of the training procedure, the agent can be trained to perform a new task, by using only a single demonstration of the new task. The approach developed specifically in this paper is mainly applicable to robotic tasks, and its novelty lies in the fact that the expert demonstrations are presented as only images, and thus the agent learns from raw pixels. The developed method combines the paradigm of meta-learning with imitation. Thus, in this chapter, the meta-learning framework is first introduced, followed by its integration into the imitation learning framework.

5.1 Model-Agnostic Meta-Learning (MAML)

Model-agnostic meta-learning (MAML) is not a contribution of the study in question. It was developed previously in [36] in a general machine learning setting, and is extended to imitation learning in [2]. By training for adaptation across tasks, meta-learning essentially treats each learned task as a datapoint in the parameter space. At the end of the training procedure, the learned parameters are always only one update away from the best parameters for any of the learned tasks.

The way MAML [2] works is by learning the parameters θ of a model f_θ such that new tasks, which are drawn from a distribution over tasks $P(\mathcal{T})$, can be quickly adapted to by performing simple gradient descent using only one demonstration of the new task. The model parameters are obtained by training based on the performance of $f_(\theta'_i)$ with respect to θ across tasks sampled from $P(\mathcal{T})$ as

$$\min_\theta \sum_{\mathcal{T}_i \sim P(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)})$$

where $\mathcal{L}_i$ is the task-specific loss function used to quantify the quality of the prediction on task i. This equation is also known as the meta-objective. This results in a final parameter vector which is close to the optimal parameters of all the tasks $\mathcal{T}_i \sim p(\mathcal{T})$. When faced with a new task, the parameters θ_i' for this new task are computed via a single-step of gradient descent, using a single observation of this task, as

$$\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$$

5.2 Meta-Imitation Learning Using MAML

The agent's policy $\pi : s \mapsto \hat{a}$ maps states to predicted actions. Unlike the EM-MLIRL method in the previous chapter, the demonstrations for each task are assumed to be labeled in the assumed setting. Each imitation task is defined as $\mathcal{T}_i = \{\tau = (s_1, a_1, \ldots, s_T, a_T) \sim \pi^*, \mathcal{L}(a_{1:T}, \hat{a}_{1:T}), \mathrm{T}\}$ consisting of demonstration data τ (sequence of state-action pairs) assumed to be generated by the expert policy π^*, a task-specific loss function $\mathcal{L}(a_1, \ldots a_T, \hat{a}_1, \ldots, \hat{a}_\mathrm{T}) \mapsto \mathbb{R}$ which measures the correctness/quality of the expert actions. There is a distribution over all tasks $P(\mathcal{T})$.

The entire training-testing procedure is divided into two phases—meta-training and meta-testing. During meta-training, a batch of tasks is sampled from $P(\mathcal{T})$ and two demonstrations of each task τ_i in the batch are sampled from π^*. The first demonstration of each task is used to compute the parameters θ_i' for that task as

$$\begin{aligned}\theta_i' &= \arg\min_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta) \\ &= \arg\min_\theta \sum_{t=1}^{T} \mathcal{L}_{\mathcal{T}_i}[f_\theta(s_t), a_t]\end{aligned}$$

The second demonstration is then used to compute the gradient of the meta-objective function, and compute the policy parameters. The error on the second demonstration according to the loss function $\mathcal{L}$ serves as the validation error to update the parameters of the agent's policy π. During meta-testing, a new task τ_j is sampled from $P(\mathcal{T})$. After fine-tuning the meta-trained model's parameters using one demonstration, the model's performance is measured on the new task. The algorithm is summarized in Algorithm 3.

5.3 Contributions and Critique

The major contribution of the work in [2] is the use of computer vision-based policies for imitation learning in a multitask setting, wherein a policy is first pre-trained on a wide-variety of tasks, followed by fine-tuning using a single demonstration of a new task. The developed approach is in many ways similar to transfer

Algorithm 3 Meta-Imitation Learning Using MAML

```
    **Input** Distribution over tasks $P(\tau)$
    **Output** Parameters $\theta$ that can be quickly adapted to new tasks through imitation
    $\alpha$, $\beta$: gradient descent step size hyperparameters
    **Initialize** Randomly initialize $\theta$
1: procedure
2:     while not done do
3:         Sample batch of tasks $\tau_i \sim P(\tau)$
4:         for all $\tau_i$ do
5:             Sample demonstration $\tau = (s_1, a_1, ..., s_T, a_T)$
6:             Evaluate $\nabla_\theta \mathcal{L}_{\tau_i}(f_\theta)$ using $\tau$ and task-specifc loss $\mathcal{L}_{\tau_i}$
7:             Compute adapted parameters with gradient descent: $\theta'_i = \theta - \alpha \nabla_\theta \mathcal{L}_{\tau_i}(f_\theta)$
8:             Sample demonstration $\tau_i = (s'_1, a_1, ..., s'_T, a'_T)$ from $\tau_i$ for meta-upfate
9:         Update $\theta = \theta - \beta \nabla_\theta \sum_{\tau_i \sim \tau} \mathcal{L}_{\tau_i} f(\theta'_i)$ using each $\tau'_i$ and $\mathcal{L}_{\tau_i}$
```

learning [37] which aims to transfer knowledge from previously-learned tasks to a new one. Where the approach of [2] differs from traditional transfer learning is that the pre-training phase of their algorithm (meta-learning) only uses a two demonstrations of each task to learn the parameters of the prediction model; even in the fine-tuning (meta-testing) phase of their approach, only a single demonstration of a new task is used to adapt the parameters of the model to the new task.

Additionally, unlike the MLIRL approach in [1], the one-shot imitation approach is not restricted to fixed number of tasks that can be learned. The method allows for demonstrations from any number of tasks to be used during meta-training. While one could posit that the meta-testing performance would only improve with the number of tasks used during meta-training, the authors neither claim this, nor provide any evidence to suggest that this is the case. In fact, one could also make the counter-argument that using an inordinately high number of tasks for meta-training might actually lead to worse performance during meta-testing due to the policy parameters not having been optimized well for any task.

Finally, the one-shot imitation method necessitates the a priori availability of all task demonstrations; the method cannot be trained with demonstrations of tasks which are received sequentially. Indeed, in this situation, the availability of new demonstrations for meta-training would require re-training the policy using all of the old task demonstrations, in addition to newly-acquired ones.

Chapter 6

Lifelong Inverse Reinforcement Learning [3]

The two approaches discussed thus far have some shortcomings. EMIRL [1] is restricted to fixed number of tasks, and does not really describe how the policy parameters for a particular task must be updated when new demonstrations are added to a cluster post-training. One-shot imitation [2] on the other hand requires a wide variety of tasks during meta-training to be able to generalize well to new tasks during meta-testing. The final multitask imitation approach discussed in this study has its origins in the lifelong learning paradigm. Specifically, in this approach, an agent is shown demonstrations for multiple different tasks sequentially, and is expected to optimize over its overall performance on all the tasks it seen until that time. The algorithm also forces the agent to make use of previously-seen tasks by leveraging knowledge of previously-learned tasks to optimize the current task. Revisiting the example in chapter 1 of this study, an agent dropping a glass utensil must be negatively rewarded and consequently avoided, whether it be while loading a dishwasher, or setting up a dining table, which points to a transfer of knowledge between tasks. Lifelong IRL [3] achieves exactly that. Since the developed method is based on maximum entropy IRL [27], its overview is provided in the next chapter, before detailing the lifelong IRL method in the chapter following that.

6.1 Maximum Entropy Inverse Reinforcement Learning (MaxEnt IRL)

Given an $MDP\backslash r$ and a set of demonstrations $\mathcal{D} = \tau_1, \tau_2, \ldots, \tau_n$, each state s_t in a demonstration is represented by a d-dimensional feature vector $\phi_{s_t} \in \mathbb{R}^d$. Based on these features, each demonstration has τ_i yields a feature count $\phi_{\mathcal{T}_i} = \sum_{t=0}^{T} \gamma^t \phi_{s_t}$. An empirical expected feature count across all demonstrations can then be calculated as $\hat{\phi} = \frac{1}{n} \sum_{i=1}^{n} \phi_{\mathcal{T}_i}$. The agent's policy must match the expert's expected feature count to satisfy the condition $V^{\pi^*} = V^{\hat{\pi}^*}$. As noted in previous chapters, the reward function is a linear function,

Algorithm 4 Efficient Lifelong IRL (ELIRL)

```
    Input Size of task-specific coefficients $k$, regularization tuning parameters $\lambda$, $\mu$
1: procedure
2:     while some task $\mathcal{T}^{(t)}$ is available do
3:         $\mathcal{Z}^{(t)} \leftarrow getExampleTrajectories(\mathcal{T}^{(t)})$
4:         $\alpha^{(t)}, H^{(t)} \leftarrow inverseReinforcementLearner(\mathcal{Z}^{(t)})$
5:         $s^{(t)} \leftarrow \arg\min_{s^{(t)}} \left[ (\alpha^{(t)} - Ls^{(t)})^T H^{(t)} (\alpha^{(t)} - Ls^{(t)}) + \mu \parallel s^{(t)} \parallel_1 \right]$
6:         $L \leftarrow updateL(L, s^{(t)}, \alpha^{(t)}, H^{(t)}, \lambda)$
```

parameterized by the weight vector $\theta \in \mathbb{R}^d$ as $r_{s_t} = \theta^T \phi_{s_t}$. The cumulative reward of any demonstration τ_i is

$$\begin{aligned} r_{\tau_i} &= \sum_{s_t \in \tau_i} \gamma^t \theta^T \phi_{s_t} \\ &= \theta^T \sum_{s_t \in \tau_i} \gamma^t \phi_{s_t} \\ &= \theta^T \phi_{\tau_i} \end{aligned}$$

The algorithm makes the assumption that the expert follows a maximum entropy policy which exponentially prefers demonstrations with higher rewards; due to this the probability of observing a demonstration is then given as $P(\tau_i|\theta, T) \approx \frac{1}{Z(\theta,T)} \exp(r_{\tau_i}) \prod_{(s_t, a_t, s_{t+1}) \in \tau_i} T(s_{t+1}|s_t, a_t)$, where $Z(\theta, T)$ is the partition function. The parameters of the expert's unknown reward function are then recovered by maximizing the likelihood of the expert's trajectories under the maximum entropy distribution as

$$\theta^* = \arg\max_{\theta} \sum_{\tau_i \in \mathcal{D}} \log\left[P(\tau_i|\theta, T)\right]$$

6.2 Lifelong IRL

The lifelong IRL setup consists of a set of tasks $\mathcal{T}_1, \ldots, \mathcal{T}_{N_{max}}$ which the agent observes in that sequence. Each task $\mathcal{T}_i$ is defined by an $MDP \backslash r\mathcal{T}^{(t)} = \langle \mathcal{S}^{(t)}, \mathcal{A}^{(t)}, T^{(t)}, \gamma^{(t)} \rangle$ The agent is required to learn N_{max} reward functions $\mathcal{R} = \{r(\theta^1), r(\theta^2), \ldots, r(\theta^{N_{max}})\}$ which correspond to a set of N_{max} parameter vector $\theta = \theta^1, \theta^2, \ldots, \theta^{N_{max}}$. After observing $N(1 \le N \le N_{max})$ tasks, the agent must maximize the likelihood of having observed all the demonstrations for all the tasks observed thus far as

$$\max_{r^{(1)}, \ldots, r^{(N)}} P\big(r^{(1)}, r^{(N)}\big) \prod_{t=1}^{N} \left[\prod_{j=1}^{n_t} P(\tau_j | r^{(t)}) \right]^{\frac{1}{n_t}}$$

where $P(r^{(1)}, \ldots, r^{(N)})$ is the prior distribution on the rewards, and n_t is the number of expert demonstrations shown to the agent for task $\mathcal{T}_t$.

The key contribution of the authors in study in question is to assume that the reward functions $t^{(t)}$ for each task are related via a latent basis reward components L, which can be then used to reconstruct the true reward functions using a linear combination of these components and a sparse, task-specific coefficient vector $s^{(t)}$. The sharing of components of L between reward functions, and forcing the coefficient vectors to be sparse facilitates the transfer the of knowledge between different tasks [**?**]. The task-specific reward function parameters $\theta^{(t)} \in \mathbb{R}^d$ can be factorized into a linear combination as

$$\theta^{(t)} = Ls^{(t)}$$

where the matrix $L \in \mathbb{R}^{d \times k}$ represents a set of k latent reward vectors, and $s^{(t)}$ are the task-specific, sparse coefficient vectors. This formulation facilitates knowledge transfer between the parametric models. Placing a Laplace prior on $s^{(t)}$ (for sparsity), a Gaussian prior on L (for complexity), then plugging in the reward prior into the MaxEnt log-likelihood equation, taking logs, and re-arranging the terms the following equation is obtained

$$\min_L \frac{1}{N} \sum_{t=1}^{N} \min_{s^{(t)}} \left\{ -\frac{1}{n_t} \sum_{\tau_i^{(t)} \in \mathcal{D}^{(t)}} \log \left[P(\tau_i^{(t)} | L_{s_i}^{(t)}, T^{(t)}) \right] + \mu \parallel s^{(t)} \parallel_1 \right\} + \lambda \parallel L \parallel_F^2$$

This equation, which is separably convex in L and $s^{(t)}$, is solved by the authors in [3] by adopting approaches from lifelong learning literature [38]. Taking the second-order Taylor expansion around a point estimate of $\alpha^t = \arg\min_\alpha - \sum_{\tau_i^{(t)} \in \mathcal{D}^{(t)}} \log \left[P(\tau_i^{(t)} | \alpha, T^{(t)}) \right]$, substituting in the previous equation and simplifying gives the following formulation

$$\min_L \frac{1}{N} \sum_{t=1}^{N} \min_{s^{(t)}} \left[(\alpha^{(t)} - Ls^{(t)})^T H^{(t)} (\alpha^{(t)} - Ls^{(t)}) + \mu \parallel s^{(t)} \parallel_1 \right] + \lambda \parallel L \parallel_F^2$$

where $H^{(t)}$ is the Hessian of the MaxEnt negative log-likelihood, and is given as

$$H^{(t)} = \frac{1}{n_t} \nabla^2_{\theta,\theta} \mathcal{L}[r(Ls^{(t)}), \mathcal{D}^{(t)}]$$

For each task $\mathcal{T}_i$ provided sequentially, the equation above is approximated as a series of updates on $s^{(t)}$ and $H^{(t)}$

$$s^{(t)} \leftarrow \arg\min_s \ell(L_N, s, \alpha^{(t)}, H^{(t)})$$

$$L_{N+1} \leftarrow \arg\min_{L} \lambda \parallel L \parallel_F^2 + \frac{1}{N} \sum_{t=1}^{N} \ell(L, s, \alpha^{(t)}, H^{(t)})$$

where $\ell(L, s, \alpha^{(t)}, H^{(t)}) = \mu \parallel s^{(t)} \parallel_1 + (\alpha^{(t)} - Ls^{(t)})^T H^{(t)} (\alpha^{(t)} - Ls^{(t)})$. The entire algorithm is provided in Algorithm 4.

6.3 Contributions and Critique

The foremost contribution of the work in [3] is the transferability of knowledge between tasks. This differs from the one-shot imitation [2] formulation where a parameter vector is meta-learned over all tasks, and then fine-tuned to new tasks each time; in the lifelong setting, a latent basis of reward vectors is learned which is truly shared between all tasks—each task is a linear combination of these reward vectors. Additionally, each new task encountered in the training sequence updates the components of the basis L. This leads to the re-use of previously-learned skills for new tasks.

The EMIRL and one-shot imitation approaches discussed in the previous chapters are limited by the fact that they require all the training demonstrations to be available a priori; EMIRL additionally also requires the number of tasks represented in the demonstrations to be known a priori. This is not the case with lifelong IRL which is explicitly formulated to deal with the sequential arrival of different task demonstrations. The downside to the sequential arrival of tasks is the following. The shared knowledge in L is incrementally refined. The task-specific coefficients $s^{(t)}$ are not updated after training on task t is complete. However, the reward parameters for this task $\theta^{(t)} = Ls^{(t)}$ may still be affected due to the changes in L due to subsequent tasks. The authors report that while such changes are typically found to improve performance of prior tasks, in certain situations, the performance might also be adversely impacted. Hence, despite the fact that the authors provide a convergence guarantees, this may not be reflect of the performance at all. One rather inelegant solution to this problem that the authors provide is to repeat the optimization on $s^{(t)}$ before testing on a task t in order to account for changes that may have occurred in L.

Of the three multitask imitation learning techniques reviewed in this study, the formulation of the lifelong IRL technique [3] may be the most versatile; indeed, it is not restricted by the number of tasks or the prior unavailability of demonstration data. However, the major drawback detailed in the previous paragraph situates the technique closer to the one-shot imitation technique, where fine-tuning is required before testing on a new task. In fact, while the fine-tuning in one-shot imitation allows it to test on previously-unseen tasks, the fine-tuning in lifelong IRL may be required to perform well on previously-seen tasks.

Chapter 7

Future Work

Multitask imitation learning carries tremendous potential for application in a multitude of different research areas. Although the field has been in development for almost a decade, examples of its applications to real-world scenarios are still few and far between. One obvious application of these techniques is robotics, where existing machine learning techniques technologies have already enabled robots to achieve human-level performance on single tasks. Enabling robots to learn multiple tasks to make them more versatile seems like the obvious next step. For example, consider two commonly-studied robotic tasks- pick-and-place and turning a door knob. While exceptional performance has been achieved on both of these tasks separately, multitask imitation could allow a single agent to learn both.

Moving forward, the nature of the application will dictate the type of multitask imitation learning to be used. For example, in the example from the previous paragraph, where the two tasks share a common state-action space, techniques such as EMIRL could be employed. For an an agent to learn a more exotic variety of tasks (such as learning to both, fold a piece of clothing and to untangle a rope) more sophisticated techniques such as lifelong IRL will have to be employed, given that such tasks do not share a common state-action space. Additionally, lifelong IRL lends itself to more complicated tasks such as these due to the fact that there may be knowledge (which may be hard to explicitly encode, or demonstrate) which could be learned from folding clothes, which could help the agent untangle the rope.

On the research side of things in multitask IRL, there has not been any groundbreaking progress that has been made in the last two years, although there are some interesting problems which need addressing. For instance, while lifelong IRL can be very versatile, it suffers from the drawback that the task-specific coefficients need updating to account for any changes in the basis reward vectors that may have occured. This is due the fact that all demonsrations for a particular task are observed sequentially by the agent not allowing the coefficients for that task to be updated. A simple workaround to this problem could be to sample batches of demonstrations for each task, and provide them to the agent in a shuffled manner. Thus a coefficients $s^{(t)}$ for task first encountered in the early stages of training have the opportunity to update again when another batch of demonstrations for task t is encountered again during later stages of training.

Chapter 8

Conclusion

This study takes a closer look at multitask imitation learning and its evolution over the last decade. Specifically, three techniques achieving this are detailed and critiqued. These techniques vary vastly in their approach to multitask imitation. It is found that the applicability of these techniques varies depending on the nature of the application. For simpler tasks which share state-action spaces, an unsupervised EMIRL approach may be conducive, whereas for a set of tasks which are more complicated and whose reward functions could benefit from shared knowledge, a lifelong IRL approach is recommended. Also presented in this study is a proposed extension to the lifelong IRL approach to improve its effectiveness and versatility. Specifically, it is recommended that demonstrations for different tasks be presented to the agent in batches, and in a shuffled order to promote better exchange of knowledge between different tasks.

References

[1] Monica Babes, Vukosi Marivate, Kaushik Subramanian, and Michael L. Littman. Apprenticeship learning about multiple intentions. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 897–904, 2011.

[2] Chelsea Finn, Tianhe Yu, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. One-shot visual imitation learning via meta-learning. *arXiv preprint arXiv:1709.04905*, 2017.

[3] Jorge Mendez, Shashank Shivkumar, and Eric Eaton. Lifelong inverse reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 4502–4513, 2018.

[4] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, and Georg Ostrovski. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015. Publisher: Nature Publishing Group.

[5] Abhishek Das, Satwik Kottur, José MF Moura, Stefan Lee, and Dhruv Batra. Learning cooperative visual dialog agents with deep reinforcement learning. In *Proceedings of the IEEE international conference on computer vision*, pages 2951–2960, 2017.

[6] Brenna D. Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

[7] J. Andrew Bagnell. An invitation to imitation. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA ROBOTICS INST, 2015.

[8] Stefan Schaal. Is imitation learning the route to humanoid robots? *Trends in cognitive sciences*, 3(6):233–242, 1999.

[9] Aude G. Billard, Sylvain Calinon, and Rüdiger Dillmann. Learning from humans. In *Springer handbook of robotics*, pages 1995–2014. Springer, 2016.

[10] Aude Billard and Daniel Grollman. Robot learning by demonstration. *Scholarpedia*, 8(12):3824, 2013.

[11] Aude Billard, Sylvain Calinon, Ruediger Dillmann, and Stefan Schaal. Survey: Robot programming by demonstration. *Handbook of robotics*, 59(BOOK_CHAP), 2008.

[12] Michael Bain and Claude Sammut. A Framework for Behavioural Cloning. In *Machine Intelligence 15*, pages 103–129, 1995.

[13] Stuart Russell. Learning agents for uncertain environments. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 101–103, 1998.

[14] Peter Moylan and Brian Anderson. Nonlinear regulator theory and an inverse optimal control problem. *IEEE Transactions on Automatic Control*, 18(5):460–465, 1973.

[15] Richard Serfozo. *Basics of applied stochastic processes*. Springer Science & Business Media, 2009.

[16] Richard S. Sutton and Andrew G. Barto. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.

[17] Masashi Sugiyama. *Statistical reinforcement learning: modern machine learning approaches*. Chapman and Hall/CRC, 2015.

[18] Csaba Szepesvári. Algorithms for reinforcement learning. *Synthesis lectures on artificial intelligence and machine learning*, 4(1):1–103, 2010.

[19] Takayuki Osa, Joni Pajarinen, Gerhard Neumann, J. Andrew Bagnell, Pieter Abbeel, and Jan Peters. An algorithmic perspective on imitation learning. *Foundations and Trends® in Robotics*, 7(1-2):1–179, 2018.

[20] Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. No-regret reductions for imitation learning and structured prediction. In *In AISTATS*. Citeseer, 2011.

[21] Shreyansh Daftry, J. Andrew Bagnell, and Martial Hebert. Learning transferable policies for monocular reactive mav control. In *International Symposium on Experimental Robotics*, pages 3–11. Springer, 2016.

[22] Stéphane Ross and Drew Bagnell. Efficient reductions for imitation learning. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 661–668, 2010.

[23] Alessandro Giusti, Jérôme Guzzi, Dan C. Cireşan, Fang-Lin He, Juan P. Rodríguez, Flavio Fontana, Matthias Faessler, Christian Forster, Jürgen Schmidhuber, and Gianni Di Caro. A machine learning approach to visual perception of forest trails for mobile robots. *IEEE Robotics and Automation Letters*, 1(2):661–667, 2015.

[24] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, and Jiakai Zhang. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[25] Andrew Y. Ng and Stuart J. Russell. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, pages 663–670, 2000.

[26] Nathan D. Ratliff, J. Andrew Bagnell, and Martin A. Zinkevich. Maximum margin planning. In *Proceedings of the 23rd international conference on Machine learning*, pages 729–736, 2006.

[27] Brian D. Ziebart, Andrew L. Maas, J. Andrew Bagnell, and Anind K. Dey. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.

[28] Sergey Levine, Zoran Popovic, and Vladlen Koltun. Nonlinear inverse reinforcement learning with gaussian processes. In *Advances in Neural Information Processing Systems*, pages 19–27, 2011.

[29] Markus Wulfmeier, Peter Ondruska, and Ingmar Posner. Maximum entropy deep inverse reinforcement learning. *arXiv preprint arXiv:1507.04888*, 2015.

[30] Deepak Ramachandran and Eyal Amir. Bayesian Inverse Reinforcement Learning. In *IJCAI*, volume 7, pages 2586–2591, 2007.

[31] Manuel Lopes, Francisco Melo, and Luis Montesano. Active learning for reward estimation in inverse reinforcement learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 31–46. Springer, 2009.

[32] Christos Dimitrakakis and Constantin A. Rothkopf. Bayesian multitask inverse reinforcement learning. In *European workshop on reinforcement learning*, pages 273–284. Springer, 2011.

[33] Jaedeug Choi and Kee-Eung Kim. Nonparametric Bayesian inverse reinforcement learning for multiple reward functions. In *Advances in Neural Information Processing Systems*, pages 305–313, 2012.

[34] George H. John. When the best move isn't optimal: Q-learning with exploration. In *AAAI*, page 1464. Citeseer, 1994.

[35] Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

[36] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1126–1135. JMLR. org, 2017.

[37] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009. Publisher: IEEE.

[38] Paul Ruvolo and Eric Eaton. ELLA: An efficient lifelong learning algorithm. In *International Conference on Machine Learning*, pages 507–515, 2013.