A Knows-What-It-Knows Algorithm for Inverse Reinforcement Learning

A Project Report

submitted by

VAISHNAVH N CS11B026

in partial fulfilment of the requirements

for the award of the degree of

BACHELOR OF TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY, MADRAS.

MAY 2015

THESIS CERTIFICATE

This is to certify that the thesis titled **A Knows-What-It-Knows Algorithm for Inverse Reinforcement Learning**, submitted by **Vaishnavh N CS11B026**, to the Indian Institute of Technology, Madras, for the award of the degree of **Bachelors of Technology**, is a bonafide record of the research work done by him under my supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Dr. Balaraman Ravindran
Research Guide
Associate Professor
Dept. of Computer Science and Engineering
IIT-Madras, 600 036

Place: Chennai

Date: May 2015

ACKNOWLEDGEMENTS

I wish to express my sincere thanks to **Prof. Balaraman Ravindran**, my research advisor, for his tremendously invaluable and patient guidance throughout the course of this project. The discussions I have had with Prof. Ravindran have been a constant source of encouragement and hope that helped me develop an obsession with the problem and helped me keep exploring for ideas even when all doors seemed to be closed.

I would also like to offer my special thanks to **Aishwarya Padmakumar**, (B. Tech Student, Department of Computer Science IIT Madras) for her support as a friend.

I am grateful to the members of the reviewing committee of **RLDM**, The Multi-disciplinary Conference on Reinforcement Learning and Decision Making, who provided interesting insights and suggestions with regard to an abstract of this work titled ***KWIK Inverse Reinforcement Learning***.

I wish to thank **Prof. Csaba Szepesvari**, Department of Computing Science, University of Alberta, for his suggestions and views about this work.

I wish to thank **Prof. Michael L. Littman**, Department of Computer Science, Brown University, for providing reference to a relevant work.

Finally, I wish to thank the my parents, my friends, and the faculty of Department of Computer Science, IIT Madras for their kind support and assistance.

Vaishnavh Nagarajan

ABSTRACT

KEYWORDS: Learning from Demonstrations, Imitation Learning, Inverse Reinforcement Learning, Knows-What-It-Knows Learning

Learning from demonstrations is an expert-to-learner knowledge transfer mechanism that has led to state-of-the-art performances in many practical domains such as autonomous navigation and robotic control. The aim of the learning agent is to *imitate* the expert observing its trajectories. One solution to this problem is inverse reinforcement learning (IRL), where the learner infers a reward function over the states of the Markov Decision Process on which the mentor's demonstrations seem optimal. However, since expert trajectories are not cheap, it becomes crucial to minimize the number of trajectories required to imitate accurately. Moreover, in large state spaces, the agent must generalize knowledge acquired from demonstrations covering few states to the rest of the states confidently. To address these requirements, we propose a reduction of IRL to classification where determining the separating hyperplane becomes equivalent to learning the reward function itself. Further, we also use the power of this equivalence to propose a Knows-What-It-Knows (KWIK) algorithm for IRL. To this end, we also present a novel definition of admissible KWIK classification algorithms which suit our goal. The study of IRL in the KWIK framework is of significant practical relevance primarily due to the reduction of burden on the teacher: a) A self-aware learner enables us to avoid making redundant queries and cleverly reduce the sample complexity. b) The onus is now on the learner (and no longer on the teacher) to proactively seek expert assistance and make sure that no undesirable/sub-optimal action is taken.

TABLE OF CONTENTS

List of Algorithms

Todo list

ABBREVIATIONS

RL	Reinforcement Learning
IRL	Inverse Reinforcement Learning
KWIK	Knows-What-It-Knows
PAC	Probably Approximately Correct
MB	Mistake Bound
MDP	Markov Decision Process
LfD	Learning from Demonstrations

NOTATIONS

Fill up

M	A Markov Decision Process
$\mathcal{S}$	State space of the MDP M
$\mathcal{A}$	Action space of the MDP M
$R(\cdot)$	Reward function of the MDP M
$T(\cdot)$	Transition function of the MDP M
γ	Discount factor
k	Dimensionality of the reward parametrization of MDP M
$\boldsymbol{\theta}$	A variable weight parameter
$\boldsymbol{\theta}_E$	The true (expert) weight parameter
$\hat{\boldsymbol{\theta}}$	An estimate of the weight parameter
$\mathbf{x}$	A multi-dimensional point represented as a vector
$\pi(\cdot)$	A policy on MDP M
V_M^π, V^π	Value function of policy π on MDP M
Q_M^π, Q^π	State-Action value function of policy π on MDP M
$\mathcal{D}$	Set of demonstrations on M
$Pr(\cdot)$	Probability of occurrence of event
$\mathbb{I}(\cdot)$	Indicator function
$\mathbb{E}[\cdot]$	Expectation of a random variable
z	Observation provided by the teacher
SGN$(\cdot)$, SGN$[\cdot]$	Signum function
$\mathbf{R}$	Rewards of MDP M represented as a vector $\in \mathbb{R}^{\lvert\mathcal{S}\rvert}$
$\mathbf{T}_\pi$	Transition matrix corresponding to policy π on MDP M
$\mathbf{V}^\pi$	Vector of values of states corresponding to π on MDP M
$\mathbf{d}_\pi(\cdot)$	Stationary distribution of a policy π
$\mathcal{C}$	KWIK Classification Algorithm

CHAPTER 1

INTRODUCTION

Use citations

Explain the stuff more properly

In this section, we will introduce the reader to some specific and fundamental concepts of reinforcement learning and machine learning that will be required to motivate and understand the problem. The technical preliminaries are postponed to a later section. For the sake of the reader who might be completely new to the reinforcement learning set-up, mathematical jargon will be cautiously avoided in this section.

We will begin by discussing the full reinforcement learning framework in Section 1.1 following which we will the *inverse* reinforcement learning problem in the context of imitation learning. Next we will describe the online learning framework of Knows-What-It-Knows (KWIK). Finally, we briefly describe the problem that forms the goal of this work.

1.1 Reinforcement Learning

The *full* reinforcement learning (RL) problem, which is what concerns us in this work is a way of addressing **sequential decision making problems**. These problems consist of two entities: an **environment** and an **agent**. At various time-steps during what is called an episode, the environment is found to be at different **states**. At each time-step, the agent performs a specific **action**. The action results in two consequences which are some form of feedback from the environment. First, the agent receives a **reward**. Second, the environment makes a **transition** to another state. The ‘problem’ here is that the agent is required to maximize some form of cumulative reward called as the **return**. As an example, a robot navigating through a maze is a sequential decision making problem: at various instances, the robot makes a move in a specific dirIection (the action), finds itself in various positions in the maze (the state), and receives a reward (punishment) from the maze if it reaches its goal (hits the wall).

When an agent is thus introduced to an unknown environment it has to learn through experience what is the ‘best’ action to be taken at every state. The quality of the action can be understood to correspond to the amount of return that is expected by taking that action. We will call this state-action mapping that is to be learned as the **optimal policy**. This process of learning will involve **exploration** - that enables to the agent to understand the environment better - and also **exploitation** that ensures that the agent makes best use of what it has learnt. Thus, when an agent explores and receives a high reward, the reward is a means of reinforcement which encourages the agent to believe that the steps it took recently are good and can to be exploited later.

Algorithms that solve this problem are broadly of two kinds. One class of algorithms maintain a model of the world by approximately estimating the stochastic feedback of the environment - these are called **model-based** algorithms. The other class of algorithms are **model-free** in that they do not maintain any such model but only work with some sort of estimate of the ‘value’ or the goodness of the actions that can be taken at every state.

With this basic idea of the reinforcement learning problem, we now look at imitation learning and then describe how inverse reinforcement learning is one way of solving it.

1.2 Imitation Learning

Recall that the goal of the reinforcement learning problem is to find the ‘optimal policy’ i.e., a mapping that tells us what is the best action to perform at a given state in a sequential decision making process. The imitation learning problem gives an interesting twist to this problem: we no longer have a notion of an optimal policy; instead we want to learn a policy that best mimics the behaviour of another agent whom we choose to call the **expert**.

Why is this problem relevant? Sometimes, when we want an agent to learn to perform a behaviour, we are required to set-up an environment that provides a platform for it to learn - and this implies that we will have to tune the rewards of the environment in such a way the agent learns to do exactly what we want it to do. For example, if we were to make a helicopter robot to learn to do loops in the air, we must provide reinforcement in the form of accurate rewards during each of its trials so that it will eventually converge to the correct way of performing a loop. Practically, this turns out to be a daunting task for the system designer!

Hence, imitation learning techniques are a form of automating this process: instead of designing the rewards associated with the environment and hence helping the agent learn what to do, we use expert **demonstrations** which is now the means of transferring knowledge to the agent. These demonstrations are basically trajectories of state action pairs in the environment. The language of rewards is completely absent in this knowledge transfer. Naturally, this problem is also called as **learning from demonstrations (LfD)** or **apprenticeship learning**.

Imitation learning, much like reinforcement learning itself, has been solved naturally in two ways. The **model-free** algorithms for imitation learning aim at blindly imitating the expert's action - the learner infers the expert's favorite action at a given state and simply copies that. On the other hand, **model-based** imitation learning algorithms take a less simpler path: these algorithms try to estimate a reward function on the environment! These algorithms assume that the expert behaves in such a way that it secretly optimizes some return connected to an unknown reward function of the environment. Thus, these algorithms primarily aim at inferring these rewards and then use some black-box reinforcement learning algorithm to find the policy that is optimal on these rewards. The process of inferring rewards that best explain the expert's behavior is known as inverse reinforcement learning, which is the subject of our discussion in the next section.

1.3 Inverse Reinforcement Learning

Recall that in the reinforcement learning problem, as the agent interacts with the environment it receives rewards that helps the agent discriminate between what sequence of state-action pairs is *good* and what sequence is *bad*. In inverse reinforcement learning, we no longer have immediate access to such rewards. Instead, we only have trajectories of an expert that is assumed to be optimal (or close to optimal) with respect an underlying reward system. The main goal of inverse reinforcement learning is to to understand what these rewards are the helps us tell that the expert's demonstrated behavior is better than any other possible behavior. We could later use these rewards to build our own policy that is close to that of the expert, which is what imitation learning aspires to do.

1.3.1 Generalization in IRL

One powerful aspect of inverse reinforcement learning is that it helps us **generalize** from expert demonstrations. Often rewards are represented as a function of certain attributes of the state-action pair that it corresponds to. That is each state-action pair's reward is encoded as a vector of features that describe the various characteristics of taking that action on that state. Depending on the task at hand, different weights are associated with each of these features and hence the reward corresponding to the state-action pair is a linear combination of these weights and the features of the pair.

In this setup, estimating the reward boils down to determining the parameters of this function that takes in these state-action pair attributes as input and produces the real valued reward. Thus, by estimating these parameters or the weights through demonstrations on a small part of the state-action space, we will be able to evaluate actions on states that have *not been* visited by the expert. Essentially, we will be able to learn what to do when the agent is placed in a new state where the expert did not happen to demonstrate any action. We must also note another advantage of inverse reinforcement learning: the parameters of this reward function gives us a **succinct representation** of the demonstrated trajectories.

In the following section, we focus on a completely different topic called as *Knows-What-It-Knows* learning which is a framework for online learning that we would like to use for analysing imitation learning.

1.4 Knows-What-It-Knows (KWIK) Learning

KWIK learning is basically a framework provided for designing online learning algorithms for which certain guarantees can be provided. In machine learning, a learner tries to approximate a unknown function that has been chosen by the environment. The knowledge about this function is derived through sample points that form the learner's experience. These **sample points** are pairs of inputs and outputs where the outputs tend to be noisy. Furthermore, these points are drawn from an underlying **data distribution**. An *offline* learner is given a dataset of points from which it is expected to learn an estimate of the function.

A common way of evaluating these functions is to ensure that for any underlying function, some event of failure of the algorithm has a low probability of occurring. To understand this better, let us call a **run** of the algorithm as the process of sampling some datapoints from the distribution and using the algorithm to produce an estimate. We evaluate the accuracy of the function as the expected error of this function on a point randomly drawn from the distribution. We define the **failure** of the algorithm for this run to be equivalent to achieving an accuracy below some preset **accuracy threshold**. We would like that the event that this failure happens (which depends on how the sample points happen to be drawn), is below a specific **failure threshold**. If such guarantees can be given for an algorithm, the algorithm is called as a **Probably Approximately Correct** algorithm.

KWIK is similarly a framework for the *online learning* setup. In online learning, the agent now receives input points one after the other. However, it is required to predict the outputs as it receives the points and is evaluated based on these outputs. KWIK simply requires that the algorithm achieves accuracy whenever it produces an output. However, the algorithm is not required to produce an output all the time! It can, for finitely many times, output a *don't know*, requesting the teacher for the output. Thus, the learning experience comes from these *don't knows*. The event of failure is associated with the above condition not holding good for a run: either the algorithm makes infinitely many requests, or makes an error on a point it did not query.

The KWIK framework for online learning provides a useful platform for studying reinforcement learning algorithms that are naturally online - the agent is seeing states on after the other and does not have, at one go, access to a vast dataset of experiences.

We must also dwell on how the KWIK framework is also associated with **active learning**. In active learning, we bestow a lot of power to the learner. The learner has the right to choose points from a pool of points, as to whether she wants the output corresponding to those subset of points (every output asked adds to the cost!). Active learning is relevant when asking for the output on all input points is redundant and adds to the cost of obtaining labels. KWIK happens to incorporate this active form of learning in its own way. A KWIK algorithm is **self-aware** in that it decides to query for the teacher's output on a select set of points and it is also aware that it does not need the teacher's output on the remaining points. Thus, KWIK also provides a framework for studying active learning algorithms. We thus discuss in the following section how KWIK can be used to study learning from demonstrations via inverse reinforcement learning.

1.5 KWIK Inverse Reinforcement Learning

Recall that active learning algorithms are useful when datapoints that a learner is trained on consists of many redundant input-output pairs. Hence, it would have been wiser to choose a subset of points for which we need the teacher's output so that we can take advantage of this redundancy and reduce the cost of querying. Similarly, in learning from demonstrations, we could expect many demonstrations to be redundant in that the expert demonstrates a lot of trajectories in the same region of the environment. It really does not help to ask for more trajectories in this region because we have learned as much as we could here. However, if we were to accidentally enter an unseen region in the environment, what do we do? We could ask for a demonstration! Here lies the motivation for designing an online algorithm for learning from demonstrations.

However, we must note that we might not always want a demonstration from an unseen state - what if we have seen a very similar state in the environment earlier where the expert has provided many demonstrations already? It would be redundant to ask for a demonstration in this unseen state. Hence, we would need strong active learning algorithms which will be able to understand when it would be wise to query and when it would not be.

Assuming we do not make a query to the expert, and allow the learning agent to take an action she thinks is the best from what she has seen of the teacher until now, how do we know that the action taken is ‘good enough’? This is where we make use of the KWIK guarantee which tells us that on every instance that the learner makes a prediction, the learner is nearly accurate. Since this accuracy does not translate to meaningful terms in the model-free imitation learning algorithm, we have chosen to work with inverse reinforcement learning based imitation learning, where this accuracy translates to taking an action that is almost good with respect to its return.

1.6 Rest of the document

CHAPTER 2

PRELIMINARIES

2.1 Markov Decision Processes

The environment in a reinforcement learning problem is often modeled as a **Markov Decision Process** (MDP) which provides framework for representing the sequential decisions that an agent takes in the environment, the feedback signals that the environment provides, and the states that the environment transitions to.

Definition 1 *Formally, an MDP is a five-tuple* $(\mathcal{S}, \mathcal{A}, T, R, \gamma)$ *where*

- $\mathcal{S}$ *is the set of states that the environment can be in. This set can either be discrete or continuous and can be parametrized in any form. This is formally called the* ***state space****.*
- $\mathcal{A}$*, or the* ***action space****, is the set of actions that the agent can perform.*
- T *defines the* ***transitions*** *of the model as enforced by the environment.* $\mathcal{T} \in (\mathcal{P}_\mathcal{S})^{\mathcal{S}\times\mathcal{A}}$*, i.e.,* $\mathcal{P}_\mathcal{S}$ *is a probability distribution over the states. That is, if at state* $s \in \mathcal{S}$*, the agent performs an action* $a \in \mathcal{A}$*, the agent goes to a next state* $s' \in \mathcal{S}$ *with probability defined by* $T(s'|s, a)$*.*
- $R \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$ *defines the rewards that the environment provides to the agent. That is, if at state* $s \in \mathcal{S}$*, the agent performs an action* $a \in \mathcal{A}$ *it sees a reward of* $R(s, a)$*.* R *can usually be defined as a distribution whose expected value is known.*
- $\gamma \in (0, 1)$ *is a term that determines how the cumulative sum of rewards is calculated over a long term. While we will mathematically discuss the significance of* γ *in the definition of* ***return****, for now we will claim that when* γ *is lower, the cumulative sum gives more weight to the immediate rewards.*

The **dynamics** of the MDP as implied by the above 5-tuple is such that the system proceeds in discrete **timesteps** $t = 0, 1, \ldots$. The agent is said to be at state $s_t \in \mathcal{S}$ at a given timestep t and it executes action $a_t \in \mathcal{A}$ at that state. On execution of the action, the environment provides a noisy reward whose expectation is $R(s_t, a_t)$ and takes the agent to the state $s_{t+1} \in \mathcal{S}$ which is a random variable that obeys the distribution $T(s_{t+1}|s_t, a_t)$.

The reader must note that throughout this discussion we have subtly assumed that the environment is ***markov***. That is, the environment behaves in such a way that the state and the reward signals it provides at timestep $t + 1$ is dependent only on the state and rewards at time t.

We will now define the notion of return which is used to *evaluate* the actions taken by the agent in terms of the rewards that it observes.

Definition 2 *The* ***return*** *of the agent at timestep t is defined as:*

$$R_t \overset{\text{def}}{=} \sum_{\tau=0}^{\infty} \gamma^{\tau} r_{t+\tau}$$

Observe that the return R_t is a random variable whose distribution is dependent on many factors. First, it is dependent on the actions that the agent takes at every timestep that follow t. Second, it depends on the stochasticity in the state-action transitions that is defined by the environment. Third, it is dependent on the reward distributions that are defined for every state-action pair of the environment.

2.2 Policy and Value Functions

Definition 3 *A* ***deterministic policy*** $\pi : \mathcal{S} \rightarrow \mathcal{A}$ *is a mapping from states to actions in the MDP.*

A policy of this form is the language that is used to describe the *behaviour* of an agent.

Definition 4 *The* ***stationary distribution*** $d_\pi : \mathcal{S} \rightarrow \mathbb{R}$ *of a policy* π *is a probability distribution over the states* $\mathcal{S}$ *such that* $\forall s, s' \in \mathcal{S}$*:*

$$d_\pi(s') = \sum_{s \in \mathcal{S}} d_\pi(s) T(s'|s, \pi(s)) \tag{2.1}$$

The stationary distribution of π *also denotes the following:*

$$d_\pi(s) = \lim_{t \rightarrow \infty} Pr(s_t = s|\pi) \tag{2.2}$$

where s_t *refers to the state reached at time* t *by following policy* π *on the MDP.*

We will now describe value functions which form the means of evaluating the goodness of a policy.

Definition 5 *The* ***value function*** $V^\pi_M : \mathcal{S} \rightarrow \mathbb{R}$ *of a policy* π *is defined as:*

$$V^\pi_M(s) \overset{\text{def}}{=} \mathbb{E}_\pi[R_0|s_0 = s]$$

The expectation in the above definition takes into account the randomness in the rewards and the transitions, and also the policy π that the agent follows in order to pick actions at every state.

While a value function helps us understand how rewarding a state is under a policy, we would also like to see how rewarding an action is under a policy. Thus, we would like to define what is called as the **state-action value** function.

Definition 6 *The **state-action value function** $Q^{\pi}_{M} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ of a policy π is defined as:*

$$Q^{\pi}_{M}(s, a) \overset{\text{def}}{=} \mathbb{E}_{\pi}[R_0 | s_0 = s, a_0 = a]$$

The expectation in the above definition takes into account the randomness in the rewards and the transitions, and also the policy π that the agent follows in order to pick actions at every state but the initial state s_0.

In the rest of this work, we may use Q^{π} in place of Q^{π}_{M}, whenever it is understood that we are dealing with a single MDP M.

Before proceeding to the inverse reinforcement learning problem, it is essential to understand what an optimal policy is.

Definition 7 *An **optimal policy** π^* of an MDP M is such that $\forall s \in \mathcal{S}$*

$$V^{\pi^*}_{M}(s) = \max_{\pi} V^{\pi}_{M}(s)$$

For the sake of simplicity, we will use V^* to refer to the optimal value function. We will now define near-optimality of a state-action pair. We look at this because for the main problem that we consider in this work, we would like to produce a policy that is near-optimal with respect the underlying rewards.

Definition 8 *At a state $s \in \mathcal{S}$, we say that for a policy π, action a is $\epsilon-$optimal, if:*

$$Q^{\pi}_{M}(s,a) \geq V^{*}_{M}(s) - \epsilon$$

2.2.1 Bellman Equations

While the definitions for the value functions provided a closed form representation, we will now present equations, called as the Bellman Equation, that related the value functions for one state to the other.

$$\begin{aligned} V^{\pi}_{M}(s) &= R(s,\pi(s)) + \gamma \sum_{s' \in \mathcal{S}} T(s'|s,\pi(s)) V^{\pi}_{M}(s') \\ Q^{\pi}_{M}(s,a) &= R(s,a) + \gamma \sum_{s'} T(s'|s,\pi(s)) \max_{a' \in \mathcal{A}} Q^{\pi}_{M}(s',a') \end{aligned} \tag{2.3}$$

2.3 Parametrized Reward/Value Functions

In Section 1.3.1, we discussed how rewards that are specified as a linear combination of features helps us in generalizing. We formally describe the set up here.

We assume that we are given the $k-$dimensional feature encoding of the *reward* on all state-action pairs as $\mathbf{\Phi}_R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^k$. A given task is specified by an underlying weight vector $\boldsymbol{\theta}$ such that $R(s,a) = \boldsymbol{\theta}^T \Phi_R(s,a)$, $\forall (s,a) \in \mathcal{S} \times \mathcal{A}$.

We must note that for any policy π, the value functions can similarly be parametrized as it is a linear combination of these rewards.

$$\begin{aligned} Q_M^\pi(s,a) &= \mathbb{E}_\pi[\textstyle\sum_{t=0}^\infty \gamma^t R(s_t,a_t)|s_0=s,a_0=a] \\ &= \mathbb{E}_\pi[\textstyle\sum_{t=0}^\infty \gamma^t \boldsymbol{\theta}^T \boldsymbol{\Phi}_R(s_t,a_t)|s_0=s,a_0=a] \\ &= \boldsymbol{\theta}^T \mathbb{E}_\pi[\textstyle\sum_{t=0}^\infty \gamma^t \boldsymbol{\Phi}_R(s_t,a_t)|s_0=s,a_0=a] \\ &\overset{\text{def}}{=} \boldsymbol{\theta}^T \boldsymbol{\Phi}_Q^\pi(s,a) \end{aligned} \tag{2.4}$$

2.4 Inverse Reinforcement Learning

Definition 9 *A demonstration d can either be a state-action pair (s,a) or a sequence of state-action pairs $(s_1,a_1),(s_2,a_2),(s_3,a_3)\ldots(s_n,a_n)$ generated by a trajectory in the MDP. That is, the sequence obeys that transition function of the MDP.*

Throughout our discussion we will refer to a state-action pair as a demonstration.

We are now ready to describe the inverse reinforcement learning problem.

Definition 10 *Given an MDP without the reward functions $M\backslash R$ i.e., $(\mathcal{S},\mathcal{A},T,\gamma)$ and a set of demonstrations of the expert $\mathcal{D}$ which usually is a set of sequences of state-action pairs, the objective of* ***inverse reinforcement learning*** *is to estimate the reward function as $\hat{R}$ such that some function of $\hat{R}$, $M\backslash R$ and $\mathcal{D}$ is optimized. The objective function usually corresponds to the likelihood of observing $\mathcal{D}$ given the parameters of $\hat{R}$ and $M\backslash R$.*

2.4.1 Problem of Multiple Optimal Solutions

A challenge in inverse reinforcement learning is that often there are many solutions to the reward functions that will explain the demonstrations by the expert that have been so far - for example, a reward setting of zero on all actions will always make the expert demonstrations optimal! However, not all these solutions are 'meaningful' and help us generalize meaningfully. This issue is addressed in many ways in literature.

One way this can be tackled is by **adding more constraints** to the optimization problem that solves the rewards. These constraints mostly necessitate the solution to provide more guarantees with regard to the demonstrations that have been made. For example, the rewards must be such that every demonstration is near-optimal.

Another way this is solved is by enforcing structural constraints on the reward function such as the inearly parameterized rewards that was discussed in the previous section. Another solution would be to *regularize* the rewards and ensure that the magnitude of the rewards is within certain bounds.

2.5 Knows-What-It-Knows Learning Framework

We will now define the problem and learning protocol for *Knows-What-It-Knows* learning. A KWIK algorithm is an algorithm that abides by this protocol for learning. The definition of the protocol inherently provides a way of evaluating these algorithms by way of knowing the guarantees these algorithms provide.

Definition 11 *A* ***KWIK problem*** *is a 5-tuple* $(\mathcal{X}, \mathcal{Y}, \mathcal{Z}, \mathcal{H}, h^*)$ *where*

- *$\mathcal{X}$ is the input space*
- *$\mathcal{Y}$ is the output space*
- *$\mathcal{Z}$ is the set of observations*
- *$\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$ is the hypothesis space*
- *$h^* \in \mathcal{Y}^{\mathcal{X}}$ is an unknown **target** function*

For theoretical purposes, we assume that the problem is ***realizable*** in that the target function belongs to our hypothesis set i.e., $h^* \in \mathcal{H}$. We will now define the KWIK protocol.

Definition 12 *The **KWIK protocol** consists of a **learner** and an adversarial **environment**. The protocol defines a **run** as follows:*

- *$\mathcal{X}, \mathcal{Y}, \mathcal{Z}, \mathcal{H}$ are all agreed upon by both the learner and the environment. Both the entities are also aware of two parameters: $\epsilon \in (0, 1)$, the accuracy parameter, and $\delta \in (0, 1)$, the confidence parameter.*
- *The environment chooses $h^* \in \mathcal{H}$ adversarially.*
- *The protocol now proceeds in timesteps, and for each timestep t:*
  - *The adversarial environment picks an input $x_t \in \mathcal{X}$ and informs the learner.*
  - *The learner predicts either a **valid output** $\hat{y}_t \in \mathcal{Y}$ or produces a $\perp$ (**don't know**).*
  - *If the learner's output is $\perp$, the environment allows the learner to observe $z_t \in \mathcal{Z}$. The KWIK protocol does not specify a relation between z_t and $y_t = h^*(x_t)$ but it is assumed that they are dependent in such a way that learning is facilitated. For example $\mathbb{E}[Z_t] = y_t$ where Z_t is a random variable whose realization is z_t.*

Add image

In our discussion, we will use the words '***environment***' and ' ***teacher***' interchangeably. This is because the environment is our source of learning as it provides us observations.

We now define KWIK-learnability, a notion that helps us in defining a class of problems that are KWIK-learnable for which algorithms with appropriate guarantees can be provided.

Definition 13 *We say that given the problem* $(\mathcal{X}, \mathcal{Y}, \mathcal{Z}, \mathcal{H}, h^*)$*,* $\mathcal{H}$ *is* ***KWIK-learnable*** *if there exists an algorithm* ***A*** *such that for any* $0 < \epsilon, \delta < 1$*, the algorithm satisfies the following two conditions with at least probability* $1-\delta$ *for any KWIK run of the algorithm:*

- ***Accuracy Condition****: If* $\hat{y}_t \neq \perp$*, the output but be* ϵ*-accurate.*
- ***Sample Complexity****: The number of* $\perp$ *produced by the learner during a run must be upper bounded by a function* $B(\epsilon, \delta, dim(\mathcal{H}))$ *-* ***the KWIK bound*** *- which is function of* 1ϵ*,* $1/\delta$*, and* $dim(\mathcal{H})$ *where* $dim(\mathcal{H})$ *is an evaluation of the dimensionality of the hypothesis space. When* $B(\epsilon, \delta, dim(\mathcal{H}))$ *is polynomial in terms of the three parameters then we call the hypothesis class* $\mathcal{H}$ ***efficiently KWIK-learnable****.*

CHAPTER 3

RELATED WORK

Give an introduction to how this is organized

3.1 Related Learning Models

In order to appreciate the features and restrictions of KWIK better, it would be worthwhile to familiarize the reader with two other learning models: the Probably Approximately Correct learning framework (PAC) Valiant [1984], and the mistake bound framework (MB) Littlestone [1987].

3.1.1 Probably Approximately Correct Learning

Go through my reports for other citations

As described in a previous chapter, the PAC model does not apply to online learning; when the learner is provided a *batch* of data offline, we would be interested in its PAC guarantees. Crucial to the PAC framework is the underlying unknown distribution $\mathcal{D}$ from which training and testing sample points are drawn independently. Let h^* be the target function chosen from the hypothesis space $\mathcal{H}$. In the training phase, a set of points are drawn $i.i.d$ from $\mathcal{X} \times \mathcal{Y}$, where $\mathcal{X}$ and $\mathcal{Y}$ are the input and output spaces respectively. The algorithm learns from this a hypothesis $\hat{h}$. We require that with probability *at least*

$1-\delta$, this hypothesis is $\epsilon-$accurate on the distribution $\mathcal{D}$. By $\epsilon-$accurate we require that the probability of the prediction going wrong on an input point drawn from $\mathcal{D}$ is less than ϵ.

$$\mathbb{E}_{x\sim D}\left[\mathbb{I}(h^*(x)\neq\hat{h}(x))\right]\leq\epsilon \tag{3.1}$$

Thus, the requirement of an $\epsilon,\delta-$ PAC learner can be formally written as, $\forall h^*$, for *any* run of the algorithm, the learner must learn $\hat{h}$ such that:

$$Pr\left(\mathbb{E}_{x\sim D}\left[\mathbb{I}(h^*(x)\neq\hat{h}(x))\right]\leq\epsilon\right)\geq 1-\delta \tag{3.2}$$

Note that we use $h^*(x)\neq\hat{h}(x)$, which might be rational to apply on discrete-valued output space. If the function has a continuous output, we might want to adjust this requirement accordingly. Nevertheless, the spirit of this definition is that the error is *not pointwise*, but is averaged over the input space according to the distribution.

One must understand the dynamics that control the failure probability δ. There are two factors that might force the algorithm to fail. First, it might be some randomness in the algorithm itself that might result in it learning a bad function. Second, it might be misfortune in the sampling phase which resulted in a bad representative set of points being sampled from $\mathcal{D}$ on which the algorithm is trained. However, we know that when sufficient number of points are sampled, this 'misfortune' is very unlikely to occur.

3.1.2 Mistake Bound Model

The mistake bound model, like KWIK, is appropriate for online learning. The difference here is that the learner is not required to *self-aware* i.e., aware of its mistakes. In KWIK, recall that whenever the learner has to predict it must be *completely sure* that it is going to be $\epsilon-$accurate on the point prediction. However, in the MB model we do not enforce the learner to know that it is making a mistake. The learner makes a prediction and the teacher provides observations on its own in order to correct the learner whenever it goes wrong. The guarantee that the algorithm must provide is that it must make a small, finite number of mistakes (i.e., $\sum_{t=1}^{\infty} \mathbb{I}(\hat{y} \neq y)$) throughout any run of the algorithm for any target function chosen adversarially.

Image

3.1.3 KWIK-MB learning

From the work of Littlestone [1989] it can be understood that MB learning is harder than PAC; and similarly from the work of Blum [1994], we understand that KWIK learning is harder than MB itself. Sayedi *et al.* [2010] have studied a model called as KWIK-MB which lies between the latter spectrum ranging from MB to KWIK learning models. The KWIK-MB learner has the privilege of making mistakes and also producing *don't knows*. That is, whenever the learner makes a prediction it can so happen that it goes wrong - the teacher interferes to correct it appropriately. However, this can happen only finitely many times upper-bounded by a parameter k. In order to make sure that it does not make too many mistakes, the learner has to carefully withdraw from making a prediction whenever it can; and this is when it outputs a $\perp$.

The beauty of this model is that, while the above description gives the perspective that the $\perp$ outputs are a refuge from making too many mistakes, one may also look at it the other way round. The learner, in order to avoid producing too many *don't know*s decides to boldly make a prediction at times. The number of times this prediction can go wrong is at most k. Hence, the mistakes can be seen as a relaxation on the KWIK-restriction.

Sayedi *et al.* [2010] have also provided an analysis of learning a linear separator in the KWIK-MB model. The algorithm they propose assumes that the points of the two classes are inherently separated by a width of γ and the KWIK-bound and itself depends on this value. We will later see how our proposed KWIK-classification algorithm can be compared and contrasted with the assumptions of this algorithm.

Why classification? Motivate.

3.1.4 KWIK online linear regression

We will now discuss the work of Strehl and Littman [2007] who have presented a KWIK online regression algorithm. Understanding the KWIK linear regression algorithm gives us insights about the style of a potential KWIK classification algorithm which we will discuss in a later chapter.

Caution the reader

In the linear regression problem we assume that the input space is $\mathbb{R}^n$ and the output space is $\mathbb{R}$. For any input point $\mathbf{x} \in \mathbb{R}^n$, if an observation $z \in \mathbb{R}$ is made, we assume that $\mathbb{E}[z] = \boldsymbol{\theta}_E^T \mathbf{x}$. That is, the teacher provides a noisy output whose expectation is linearly dependent on the features of the input point.

The algorithm works in the following fashion: at any timestep, we know that if the learner produces a *don't know* ($\perp$), the teacher provides learning experience by providing an observation of the output. Thus, before deciding on what to do, the learner uses the aggregate of all learning experiences it has had so far, and evaluates how 'far' away these experiences are from the current input point. If the experiences are sufficiently close to the input, we know that if we were to predict from the experience, we would not make a large error.

At any timestep t of the algorithm, assume that the learner would have acquired learning experience from $m < t$ points corresponding to some m timesteps in which it produced a $\perp$ so far. Let $X_t \in \mathbb{R}^{m \times n}$ denote these m input points, and $\mathbf{z} \in \mathbb{R}^m$ denote the corresponding m observations of the teacher. We can apply singular value decomposition to $X^T X$ as it is symmetric and positive semi-definite.

$$X^T X = U \Lambda U^T \tag{3.3}$$

Here $U \in \mathbb{R}^{n \times n}$ and consists of $\mathbf{u}_1, \mathbf{u}_2 \dots \mathbf{u}_n$ column vectors. Also, Λ is a diagonal matrix containing the eigenvalues $\lambda_1, \lambda_2 \dots \lambda_k$. Let $\lambda_1 \geq \lambda_2 \dots \geq \lambda_k \geq 1 \geq \lambda_k + 1 \geq \dots \lambda_n \geq 0$. We will now consider only those eigenvectors that have an eigenvalue above 1 i.e., the vectors corresponding to $\lambda_1, \lambda_2 \dots \lambda_k$. Let $\bar{U} = [\mathbf{u}_1, \mathbf{u}_2, \dots \mathbf{u}_n]$ and $\bar{\Lambda} = \text{diag}(\lambda_1, \lambda_2, \dots \lambda_k)$.

At timestep t when the learner faces the new input $\mathbf{x}_t$, the algorithm now evaluates the 'closeness' of the experience using two parameters:

$$\tilde{\mathbf{q}} \overset{\text{def}}{=} X \bar{U} \bar{\Lambda}^{-1} \bar{U}^T \mathbf{x}_t \tag{3.4}$$

$$\tilde{\mathbf{u}} \overset{\text{def}}{=} [0, \ldots, 0, \mathbf{u}_{k+1}^T \mathbf{x}_t, \ldots \mathbf{u}_n^T \mathbf{x}_t] \tag{3.5}$$

Note that $\bar{U} \in \mathbb{R}^{n \times k}$, $\bar{\Lambda} \in \mathbb{R}^{k \times k}$, $\tilde{\mathbf{q}}$.

The decision of whether a *don't know* is produced or not depends on whether $||\tilde{\mathbf{q}}||_2 > \alpha_1$ and $||\tilde{\mathbf{u}}||_2 > \alpha_2$ where α_1, α_2 are appropriately chosen constants. The algorithm is formally presented below.

Algorithm 1 KWIK Online Linear Regression

```
Require: Problem $P = (\mathcal{X}, \mathcal{Y}, \mathcal{Z}, \mathcal{H}, \boldsymbol{\theta}_\mathbf{E}), \epsilon, \delta, \alpha_1, \alpha_2$
  $X = [\,]$
  $\mathbf{z} = [\,]$
  for $t = 1, 2, \ldots$ do
    Environment picks $\mathbf{x}_t \in \mathcal{X}$
    Compute $\tilde{\mathbf{u}}$, $\tilde{\mathbf{q}}$ according to Equations (3.4), eqrefeq:u
    if $||\tilde{\mathbf{q}}||_2 \leq \alpha_1$ and $||\tilde{\mathbf{u}}||_2 \leq \alpha_2$ then
      $\hat{\boldsymbol{\theta}} \leftarrow \arg\min_{\boldsymbol{\theta} \in \mathbb{R}^n : ||vec\theta||_2 \leq 1} ||X\boldsymbol{\theta} - \mathbf{z}||_2^2$
      Learner outputs $\hat{\boldsymbol{\theta}}^T \mathbf{x}_t$
    else
      Learner outputs $\perp$
      Environment produces observation $z_t$ such that $\mathbb{E}[z_t] = \boldsymbol{\theta}_\mathbf{E}{}^T \mathbf{x}_t$
      Add $\mathbf{x}_t^T$ as a new row to $X$
      Add $z_t$ as a new element to vector $\mathbf{z}$
    end if
  end for
```

3.1.5 Sketch of Theoretical Analysis

The KWIK bound of this algorithm turns out to be $\tilde{\mathcal{O}}\left(n^3/\epsilon^4\right)$ when α_1 and α_2 are set appropriately. It will be useful to understand how the algorithm is analyzed to affirm its guarantees. Li *et al.* [2011] provide a detailed proof for the correctness of the above algorithm under appropriate parameter settings.

3.1.5.1 Proof of Correctness

The essence of the proof for correctness is that, if $||\tilde{\mathbf{q}}||_2 \leq \alpha_1$ and $||\tilde{\mathbf{u}}||_2 \leq \alpha_2$, then it can ***not*** be the case that the prediction that the algorithm makes is far off from the true answer. In other words, the algorithm is ϵ-accurate whenever it makes a valid prediction.

The first step to proving this is showing that if the above condition on the parameters were true, *and if the algorithm's output is not ϵ-accurate*, then $\Delta_E(\hat{\boldsymbol{\theta}}) \overset{\text{def}}{=} \sqrt{\sum_{i=1}^{m}\left((\boldsymbol{\theta}-\hat{\boldsymbol{\theta}})^T\mathbf{x}_i\right)^2}$, the 'true' error on the training data, must be large. We call this the *true* error because this is the error of our predictions with respect to the true values of the outputs on the training data, and not with respect to the noisy observations of the training data.

The second step to the proof is showing that when this true error on the training data is sufficiently large, then the *empirical* error of the estimated parameter on the training data - which is $\sqrt{\sum_{i=1}^{m}(\hat{\boldsymbol{\theta}}^T\mathbf{x}_i - z_i)^2}$ - is larger than the empirical error of the true parameter on the training data, $\sqrt{\sum_{i=1}^{m}(\boldsymbol{\theta}^T\mathbf{x}_i - z_i)^2}$ *with high probability.* We say 'with high probability' because we use the Hoeffding bound (Lemma 3) here to make claims about the deviation of the sum of many random variables z_i from the sum of their corresponding expectations $\boldsymbol{\theta}^T\mathbf{x}_i$.

The proof is completed by claiming that when the prediction is not $\epsilon-$accurate, and hence when the true error is sufficiently large, the true parameter minimizes the empirical error better than the our own estimate. This is a contradiction because we have determined our estimate as a minimizer of the empirical error itself. Hence, to avoid this contradiction, it has to be the case that the prediction by the algorithm, when valid, has to also be $\epsilon-$accurate.

3.1.5.2 Proof for KWIK Bound

The first step towards bounding m, the number of *don't know*'s for this algorithm in terms of $n, \epsilon, \delta$is to derive a bound in terms of n, α_1, α_2. This is done using Lemma 13 of Auer [2002] which provides an upper bound on $\sum_t ||\tilde{\mathbf{q}}_t||$ and $\sum_t ||\tilde{\mathbf{u}}_t||$ in terms of m and n and by using the conditions of the algorithm which provide a lower bound on $\tilde{\mathbf{q}}_t||$ and $\sum_t ||\tilde{\mathbf{u}}_t||$ in terms of α_1, α_2.

Now, all we need to do is to determine an appropriate setting for α_1 and α_2 in terms of n, ϵ, δ such that the total failure probability is δ. This leads to the polynomial bound as has been claimed earlier. The tricky part in doing this is determining the number of times the *high-probability* condition is applied so that we can set a sufficiently small failure probability for it on using the union bound (Lemma 2). We skip the details here though they are insightful for they are not eventually relevant to the problem.

3.1.6 Selective Sampling

We discuss in this section in sufficient detail the work of Cesa-Bianchi and Orabona [2009] in what is called as *selective sampling*. Selective sampling is an approach that parallels the KWIK protocol. The online learner makes queries whenever it wants to abstain from producing a sample. However, their guarantees are not as formalized/rigid as the KWIK protocol.

A major difference between their work and in our way of defining classification is that they make stronger assumptions about the noise of the teacher - that the noise in the observations provided by the teacher is a linear function of the distance from the classifier. That is, if $\boldsymbol{\theta}_E$ was the classifier and $\mathbf{x}_t$ the point seen at time t, then the teacher's observation Y_t for the label is such that $\mathbb{E}[Y_t] = {\boldsymbol{\theta}_E}^T\mathbf{x}_t$. The actual label for the point however is $\mathrm{SGN}(\boldsymbol{\theta}^T\mathbf{x}_t)$.

The algorithm maintains an estimate of the classifer as follows:

$$\hat{\boldsymbol{\theta}} \overset{\text{def}}{=} (I + S_{t-1}S_{t-1}^T + \mathbf{x}_t\mathbf{x}_t^T)^{-1}S_{t-1}\mathbf{Y}_{t-1} \tag{3.6}$$

where:

- N_{t-1} is the number of points seen so for on which queries were made.
- $S_{t-1} = \left[\mathbf{x}'_1, \mathbf{x}'_2 \ldots \mathbf{x}_{N'_{t-1}}\right] \in \mathbb{R}^{k\times N_{t-1}}$ is a matrix containing the points on which observations were made.
- $\mathbf{Y}_{t-1} \in \mathbb{R}^{N_{t-1}\times 1}$ is a vector containing the teacher's answers to the queries made until now.

For the sake of simplicity, we will define $A_t \overset{\text{def}}{=} (I + S_{t-1}S_{t-1}^T + \mathbf{x}_t\mathbf{x}_t^T)$ We will have to define a few more terms which will be used in the conditions that will be checked for querying.

$$\begin{aligned}
B_t &= {\boldsymbol{\theta}_E}^T(I + \mathbf{x}_t\mathbf{x}_t^T)A_t^{-1}\mathbf{x_t} \\
r_t &= \mathbf{x}_t^T A_t^{-1}\mathbf{x}_t \\
\mathbf{q}_t &= S_{t-1}^T A_t^{-1}\mathbf{x}_t \\
s_t &= ||A_t^{-1}\mathbf{x}_t||_2
\end{aligned} \tag{3.7}$$

In the above equations note that B_t is an 'unknown' variable dependent on the true classifier's value.

In Equation (3.6), observe that $\hat{\boldsymbol{\theta}}_t$ is a random variable dependent on the noise of the teacher. Thus, what is the expectation of $\hat{\boldsymbol{\theta}}_t^T\mathbf{x_t}$? This forms the crux of the algorithm.

$$\begin{aligned}
\mathbb{E}[\hat{\boldsymbol{\theta}}_t^T\mathbf{x}_t)] &= \mathbb{E}[\hat{\boldsymbol{\theta}}_t^T]\mathbf{x}_t \\
&= \mathbb{E}[((I + S_{t-1}S_{t-1}^T + \mathbf{x}_t\mathbf{x}_t^T)^{-1}S_{t-1}\mathbf{Y}_{t-1})^T]\mathbf{x}_t \\
&= \mathbb{E}[\mathbf{Y}_{t-1}^T S_{t-1}^T\left((I + S_{t-1}S_{t-1}^T + \mathbf{x}_t\mathbf{x}_t^T)^{-1}\right)^T]\mathbf{x}_t \\
&= \mathbb{E}[\mathbf{Y}_{t-1}^T]S_{t-1}^T(I + S_{t-1}S_{t-1}^T + \mathbf{x}_t\mathbf{x}_t^T)^{-1}\mathbf{x}_t \\
&= {\boldsymbol{\theta}_E}^T S_{t-1}S_{t-1}^T(I + S_{t-1}S_{t-1}^T + \mathbf{x}_t\mathbf{x}_t^T)^{-1}\mathbf{x}_t \\
&= {\boldsymbol{\theta}*}^T\mathbf{x}_t - \boldsymbol{\theta}^T(I + \mathbf{x}_t^T\mathbf{x}_t)(I + S_{t-1}S_{t-1}^T + \mathbf{x}_t\mathbf{x}_t^T)^{-1}\mathbf{x}_t \\
&= {\boldsymbol{\theta}_E}^T\mathbf{x}_t - B_t
\end{aligned} \tag{3.8}$$

Thus, B_t is an unknown value that denotes the *bias* in our estimate of the point's label. We can however have upper bounds on the value of B_t through the following inequalities:

$$\begin{aligned}
|B_t| &\leq r_t + s_t \\
s_t &\leq \sqrt{r_t}
\end{aligned} \tag{3.9}$$

Now we will also need a concentration bound on the probability that $\hat{\boldsymbol{\theta}}_t^T\mathbf{x}_t$ is too far away from its expected value.

$$Pr\left(|\hat{\boldsymbol{\theta}}_t^T \mathbf{x}_t + B_t - {\boldsymbol{\theta}_E}^T \mathbf{x}_t| \geq \epsilon\right) \leq 2\exp\left(-\frac{\epsilon^2}{2||\mathbf{q}_t||_2^2}\right) \tag{3.10}$$

Thus we can need to consider the upperbound on B_t while accounting for the error in our prediction. The algorithm is given below.

Algorithm 2 Selective sampling algorithm

```
Require: Problem $P = (\mathcal{X}, \mathcal{Y}, \mathcal{Z}, \mathcal{H}, \boldsymbol{\theta}^*), \epsilon, \delta$
  $\boldsymbol{\theta} = \mathbf{0}$
  for $t = 1, 2, \ldots$ do
    Environment picks $\mathbf{x}_t \in \mathcal{X}$
    if $[\epsilon - r_t - s_T]_+ \leq ||\mathbf{q}_t||_2 \sqrt{2\ln\left(\frac{t(t+1)}{\delta}\right)}$ then
      Learner outputs $\perp$
      Environment produces observation $y_t$ such that $\mathbb{E}[y_t] = {\boldsymbol{\theta}_\mathbf{E}}^T \mathbf{x}_t$
      Update $\hat{\boldsymbol{\theta}}_t$ using Equation (3.6)
    else
      Predict SGN($\hat{\boldsymbol{\theta}}_t^T \mathbf{x}_t$)
    end if
  end for
```

Apart from making strong assumptions about the noise, the algorithm also makes **unbounded number of queries** in any run of the algorithm. The authors prove that after T steps of the algorithm, the number of queries that would have been issued is:

$$\mathcal{O}\left(\frac{k}{\epsilon^2}\left(\ln\frac{T}{\delta}\right)\ln\frac{\ln(T/\delta)}{\epsilon}\right) \tag{3.11}$$

The advantage that this algorithm however provides over the KWIK linear regression (1) is that the sample complexity varies with the dimensionality k as $\tilde{\mathcal{O}}(k/\epsilon^2)$

3.2 Inverse Reinforcement Learning

In the following section, we briefly discuss a few algorithms for inverse reinforcement learning. The aim of this discussion is to familiarize the reader with the standard approaches to solving the IRL problem. Following this section, we will survey literature on the active learning approaches to IRL, and also imitation learning in general.

3.2.1 Characterization of the Solution Set

Ng and Russell [2000] derive the necessary and sufficient condition on the reward function $R : \mathcal{S} \rightarrow \mathbb{R}$ (now represented as $\mathbf{R} \in \mathbb{R}^{|\mathcal{S}|}$) for which a given policy π will be optimal. The analysis can be extended to rewards define on the space $\mathcal{S} \times \mathcal{A}$ which we do not present here. The reader must be careful not to confuse the below discussion with the case where rewards are define for each state-action pair.

Theorem 1 *If $\mathbf{R} \in \mathbb{R}^{|\mathcal{S}|}$ the vector of rewards over the states in $\mathcal{S}$, and if $\mathbf{T}_\pi$ represents the transition matrix for any policy π, then for some π to be optimal, we require that $\forall \pi \neq \pi(s)$:*

$$(\mathbf{T}_\pi - \mathbf{T}_{\pi'})(\mathbf{I} - \gamma \mathbf{T}_\pi)^{-1}\mathbf{R} \succcurlyeq 0 \tag{3.12}$$

The above theorem can be proved by making use of the following expression for the value functions:

$$\mathbf{V}^\pi = (\mathbf{I} - \gamma \mathbf{T}_\pi)^{-1}\mathbf{R} \tag{3.13}$$

and by using the fact that $\forall \pi' \neq \pi$:

$$\mathbf{T}_{\pi}\mathbf{V}^{\pi} \succcurlyeq \mathbf{T}_{\pi'}\mathbf{V}^{\pi} \tag{3.14}$$

3.2.2 Problem of Multiple Optimal Solutions

Recollect that in Section 2.4.1, we saw that the IRL problem can have multiple solutions. For example, here $\mathbf{R} = \mathbf{0}$ belongs to the solution set of every problem! Thus, the solution set characterized as above may not necessarily be a singleton set. Ng and Russell [2000] describe constraints that can be considered in order to shrink the above solution set to a single reward vector.

One natural constraint would be to necessitate that the rewards result in a unique optimal policy. This will certainly eliminate $\mathbf{R} = \mathbf{0}$ as candidates. This might still not save us from multiple solutions. Another way to address this would be to ask for solutions that make any perturbation of the corresponding policy very bad. If π is the policy learned on $\mathbf{R}$, this requirement can be translated to maximizing:

$$\sum_{s \in \mathcal{S}} \left(Q^{\pi}(s, \pi(s)) - \max_{a \in \mathcal{A}, a \neq \pi(s)} Q^{\pi}(s, a) \right)$$

Another constraint that they discuss is adding a penalty of $\lambda ||\mathbf{R}||_1$ to the above objective which will help us regularize the solution. Thus, they pose the IRL problem as an optimization problem:

$$
\begin{aligned}
\text{maximize} \quad & \sum_{s\in\mathcal{S}} \min_{a\in\mathcal{A}, a\neq\pi(s)} (\mathbf{T}^s_\pi - \mathbf{T}^s_a)(\mathbf{I} - \gamma\mathbf{T}_\pi)^{-1}\mathbf{R} - \lambda||\mathbf{R}||_1 \\
\text{s.t.} \quad & (\mathbf{T}_\pi - \mathbf{T}_{\pi'})(\mathbf{I} - \gamma\mathbf{T}_\pi)^{-1}\mathbf{R} \succcurlyeq 0 \qquad\qquad \forall\pi' \neq \pi \\
& ||\mathbf{R}||_\infty \leq R_{max}
\end{aligned}
\tag{3.15}
$$

3.2.3 Inverse Reinforcement Learning with Parametrized Rewards

In Section 1.3.1, recall that we discussed how parametrizing the reward functions helps us in generalizing. We also formally presented this notion in Section 2.3.

Under the assumption that Φ_R is provided in the MDP model, Abbeel and Ng [2004] infer $\hat{\boldsymbol{\theta}}$ (the weight parameter for the rewards) that results in **feature expectations** close to that demonstrated by the expert.

To understand this, first note that every parameter setting $\hat{\boldsymbol{\theta}}$ (where we assume that $||\hat{\boldsymbol{\theta}}||_2 \leq 1$), corresponds to some policy $\pi(\hat{\boldsymbol{\theta}})$ computed to be optimal on the rewards $\hat{\boldsymbol{\theta}}^T R(s,a)$, $\forall(s,a) \in \mathcal{S} \times \mathcal{A}$. Next, every policy $\pi(\hat{\boldsymbol{\theta}})$ corresponds to what is called a feature expectation vector:

$$
\boldsymbol{\mu}_{\pi(\boldsymbol{\theta})} = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t \Phi_R(s_t, a_t)] \in \mathbb{R}^k \tag{3.16}
$$

Hence, every $\hat{\boldsymbol{\theta}}$ corresponds to some feature expectation vector. The feature expectation vector can be seen as the expected discounted value of the policy across each dimension of the parametrized representation of the rewards. One must also relate this term to the *true* value of a policy:

$$
\boldsymbol{\theta}_E^T \boldsymbol{\mu}_{\pi(\hat{\boldsymbol{\theta}})} = \mathbb{E}_{\pi(\hat{\boldsymbol{\theta}})}\left[V_{\pi(\boldsymbol{\theta}_E)}(s)\right] \tag{3.17}
$$

Note that throughout the discussion we will use the qualifier *true* to indicate that the quantity being referred to is the expert's. We call a parameter setting $\hat{\boldsymbol{\theta}}$ to be $\epsilon-$optimal when:

$$\mathbb{E}_{\pi(\boldsymbol{\theta}_E)}\left[V_{\pi(\boldsymbol{\theta}_E)}(s)\right] - \mathbb{E}_{\pi(\hat{\boldsymbol{\theta}})}\left[V_{\pi(\boldsymbol{\theta}_E)}(s)\right] \leq \epsilon \tag{3.18}$$

The objective of Abbeel and Ng [2004] is to find $\hat{\boldsymbol{\theta}}$ such that the resultant policy is ϵ-optimal with respect to the true parameter $\boldsymbol{\theta}_E$. This is done by ensuring that the corresponding feature expectation is $\epsilon-$approximate to the empirical feature expectation as seen from the demonstrations $\hat{\boldsymbol{\mu}}$. That is:

$$||\mu_{\pi(\hat{\boldsymbol{\theta}})} - \mu_{\pi(\boldsymbol{\theta}_E)}||_2 \leq \epsilon \tag{3.19}$$

We use $\hat{\boldsymbol{\mu}}$ as an empirical estimate of $\mu_{\pi(\boldsymbol{\theta}_E)}$. If sufficient demonstrations are seen, one can know with high probability the error in this estimate and hence can appropriately adjust the analysis to account for this error.

What the above results in is that, *whatever be the true parameter* $\boldsymbol{\theta}_E$, the policy optimal on the inferred $\hat{\boldsymbol{\theta}}$ is ϵ-optimal with respect to $\boldsymbol{\theta}_E$. This is evident from the following:

$$\begin{aligned} |\mathbb{E}_{\pi(\hat{\boldsymbol{\theta}})}[V_{\pi(\boldsymbol{\theta}_E)}(s)] - \mathbb{E}_{\pi(\boldsymbol{\theta}_E)}[V_{\pi(\boldsymbol{\theta}_E)}(s)] &= |\boldsymbol{\theta}_E^T\mu_{\pi(\hat{\boldsymbol{\theta}})} - \boldsymbol{\theta}_E^T\mu_{\pi(\boldsymbol{\theta}_E)}| \\ &\leq ||\boldsymbol{\theta}_E||_2||\mu_{\pi(\hat{\boldsymbol{\theta}})} - \mu_{\pi(\boldsymbol{\theta}_E)}||_2 \\ &\leq 1 \cdot \epsilon = \epsilon \end{aligned} \tag{3.20}$$

The algorithm is summarized below.

Algorithm 3 Apprenticeship Learning via Inverse Reinforcement Learning

```
Require: Empirical estimate of expert feature expections $\hat{\boldsymbol{\mu}}$
  $\pi_0 \leftarrow$ Randomly initialized policy
  Candidate policy pool $\Pi \leftarrow \{\pi_0\}$
  for $i = 1, 2, \ldots$ do
    $t_i \leftarrow \max_{\boldsymbol{\theta}:||\boldsymbol{\theta}||_2 \leq 1} \min_{\pi \in \Pi} \boldsymbol{\theta}^T(\hat{\boldsymbol{\mu}} - \boldsymbol{\mu}_\pi)$
    $\boldsymbol{\theta}_i \leftarrow \boldsymbol{\theta}$ that corresponds to the above maximum
    Break if $t_i \leq \epsilon$
    Compute $\pi_i$ optimal on $\boldsymbol{\theta}_i$
    $\Pi \leftarrow \Pi \cup \{\pi_i\}$
  end for
```

The algorithm returns a pool of policies such that for every setting of $\boldsymbol{\theta}_E$, some policy in the pool has an $\epsilon-$accurate feature expectation. Beyond this, Abbeel and Ng [2004] provide ways to either manually select our 'favorite' policy from the pool or automatically aggregate these.

The authors provide theoretical results for the sample complexity and the time complexity of the algorithm. The sample complexity which is of interest to us is:

$$\frac{2k}{(\epsilon(1-\gamma))^2} \ln\left(\frac{2k}{\delta}\right) \tag{3.21}$$

Though the algorithm requires only $k \ln k$ samples where k is the dimensionality of the state representation, the solution does not consider active learning approach which is the main motivation of our work.

3.2.4 Bayesian Inverse Reinforcement Learning

While the above algorithms look for a point solution to an optimization problem that models IRL, Ramachandran and Amir [2007] consider a bayesian approach to the problem i.e., can we provide a probability distribution over all possible solutions such that the distribution denotes how likely the point is the true solution?

The motivation behind Bayesian IRL is firstly the fact that a single solution to IRL is not possible without artificial constraints - which arises from a lack of information. A probability distribution over all possible solutions would hence be of interest to us.

Furthermore, in the bayesian setup, since we work with a ***posterior distribution*** (the final distribution over the solution space) that is derived from a ***prior*** (the belief over the solution space with know information) and the ***likelihood*** (the expert demonstrations), we are now in a position to include ***prior domain knowledge*** about the system.

For a specific setting of rewards $\mathbf{R} \in \mathbb{R}^{|\mathcal{S}\times\mathcal{A}|}$, the **likelihood** of the demonstrations

$$\mathcal{D} = \{(s_1, a_1), (s_2, a_2), \ldots (s_N, a_N)\}$$

which can be decomposed into state-action pairs (under the assumption that the policy is stationary) is expressed as:

$$Pr(\mathcal{D}|\mathbf{R}) = \prod_{(s,a)\in\mathcal{D}} Pr((s,a)|\mathbf{R}) \tag{3.22}$$

Now, $\forall\ (s,a) \in \mathcal{S} \times \mathcal{A}$, we can assume a dependence of $Pr((s,a)|\mathbf{R})$ on $Q_M^{\pi(\mathbf{R})}(s,a)$ where $\pi(\mathbf{R})$ is understood be the optimal policy on $\mathbf{R}$. Finally, we are interested in the following term:

$$Pr(\mathbf{R}|\mathcal{D}) = \frac{Pr(\mathcal{D}|\mathbf{R})Pr(\mathbf{R})}{Pr(\mathcal{D})} \tag{3.23}$$

3.2.5 Maximum Entropy Inverse Reinforcement Learning

When sub-optimal behaviour is demonstrated by the expert, many different distributions of policy match the feature expectation estimates that are discussed in Abbeel and Ng [2004]. Which distribution over the policies is the *best*? To determine this, Ziebart *et al.* [2008] employ the principle of maximum entropy.

Ziebart *et al.* [2008] assume a parametrization of the distribution over the *paths* from which the demonstrations have been produced. This parametrization is such that paths with equal returns are given equal probabilities and paths with greater returns are given exponentially greater probabilities. The IRL problem now boils down to determining rewards that produces a distribution over the different paths which maximizes the likelihood of the demonstrations. The technical details of this work is beyond the scope of this thesis.

3.3 Active Imitation Learning

In the following section, we will provide a complete survey of the literature on active learning for imitation learning. Recall that imitation learning as such can be approached in two different ways: the *model-free* 'supervised learning' approach and the *model-based* inverse reinforcement learning approach. We will see that heuristics have been proposed to solve either of these problems. However, theoretical analyses have been provided only for active learning in the model-free classification based approach. There has been no formal theoretical understanding on *active online algorithms* for IRL. Parallel to the discussion of the works related to our contribution, we will also discuss how our contribution differs from earlier work. Thus, the reader is strongly advised to appreciate.

3.3.1 Active Model-free Imitation Learning

Recall that in model free learning the learning agent simply tries to label states with actions based using *supervised* by the expert's demonstrations. The approach is that a multi-label classifier examines input which are state-action pairs from the demonstrations, and then generalizes this mapping across all states. This classifier could possibly a PAC classifier which provides an $(\epsilon, \delta)-$guarantee.

The traditional approach to model-free imitation learning as discussed by Syed and Schapire [2010] would be to learn a classifier over the distribution d_{π_E}. A PAC learner would provide a learned policy $\hat{\pi}$ the guarantee that with high probability $1 - \delta$, when a state is picked according to d_{π_E}, the probability of executing the wrong action is less than ϵ. However, the learning agent will be following the policy $\hat{\pi}$ which it learned. Thus, the error it will be committing while executing it going to be the probability of executing a wrong action given that the state is picked from $d_{\hat{\pi}}$. It can be shown that this discrepancy between the *training* distribution - the stationary distribution of π_E - and the *testing* distribution - the stationary distribution of the learning agent- results in a mistake of $T\epsilon^2$, where T is the episode length.

Ross and Bagnell [2010] propose the *forward training algorithm* (Algorithm 4) to address this issue. In the forward training algorithm, which is a passive imitation learning algorithm, a passive learner learns iteratively from the distribution over states that is generated by the currently learned by the policy. Remember that a passive learner requires the distribution to give $(\epsilon, \delta)-$PAC guarantees. Thus, the algorithm learns a non-stationary policy. [1] For timestep t, the forward training algorithm iteratively learns π_t from the state distribution induced by π_{t-1} with t increasing from 1 uptil T.

The authors show that the worst case guarantees for this algorithm is still the same as the traditional appraoch i.e., $T^2\epsilon$. However there are conditions where practically this algorithm outdoes the naive approach.

Algorithm 4 Forward Training Algorithm

```
Require: MDP $M$
  Initialize $\pi_1^0, \dots \pi_T^0$
  for $t$ in $0 \dots T$ do
      Sample many T-step trajectories using $\pi_1^t, \pi_2^t \dots \pi_T^t$
      Generate $\mathcal{D} = (s_i, \pi_E(s_i))$, the state action pairs seen at timestep $i$ in all the trajectories
      Train classifier on $\mathcal{D}$ to produce $\pi_i^i$
      $\forall j \neq i$, $\pi_j^i = \pi_j^{i-1}$
  end for
      return $\pi_1^T, \dots \pi_T^T$
```

Judah *et al.* [2012*a*] study the problem of active learning for model-free imitation learning. While a passive learner is dependent on the distribution in that it needs to optimize its performance with respect to the distribution, the active learner is dependent on the distribution also for the fact that it can optimize its active queries intelligently. For this reason, the authors show that using an active learner in place of a passive learning

[1]A non-stationary policy is one which follows policy π_t at timestep t.

in the conventional approach of Syed and Schapire [2010] does not help the purpose as it results in making many queries to mimic the expert's distribution for each t. On the other hand, if we were to follow the forward training approach to depend on the distribution induced by the policy learned so far we will save a number of queries. Thus, Judah *et al.* [2012*a*] show that the learner in the forward training algorithm can simply be replaced by an active learner. The active forward training algorithm is given by

$$\hat{\pi}_t = L_a(\epsilon, \frac{\delta}{T}, d^t_{\hat{\pi}_{t-1}}) \tag{3.24}$$

where L_a is an active learner.

Our work is motivated here by the fact that this algorithm applies to model-free imitation learning and is also not considered in an *interactive online* framework.

3.3.1.1 Bad State Responses

Judah *et al.* [2011] study a modified active learning model for the model-free imitation learning problem. Here, the teacher can inform the learner whether its active query is 'useless'. A query is termed useless if it is a query on a state that is never *realized*. The authors cite an example: when the agent is learning to fly a helicopter, it is not meaningful to query on the state of an unrecoverable fall. The expert advice in fact might be poor in such states. Or, the advice might be too complicated to imitate and hence we will be attempting to learn something difficult for no purpose. The authors propose that Bayesian active learning can be used to solve the problem from here. The advantage that these bad state responses gives is that according to the teacher *any* policy that it may follow these states correspond to the *same* answer ('bad'). Thus, if the learner detects that a certain state when queried will correspond to the same answer regardless of the policy of the teacher, the learner will avoid querying on this state.

3.3.1.2 Interactive Policy Learning through Confidence-Based Autonomy

Closest to our goal of actively querying for demonstrations depending on the agent's *confidence* about its inference, is the work of Chernova and Veloso [2009] and Chernova and Veloso [2007], who have explored learning from demonstrations as an ***interactive policy learning*** process where the agent takes autonomous decisions based on its confidence and depends on expert advice when it determines that its confidence is insufficient.

Chernova and Veloso [2007] use a set of Gaussian Mixture Models that help the agent model the uncertainty in the demonstrations seen so far. The GMM set provides a means for statistically analysing uncertainty and quantifying confidence. Chernova and Veloso [2009] introduce the idea of occasionally getting teacher's *corrective* advice. Thus the process of learning consists of two parts: *confidence execution* and *corrective demonstration*. In the former, the agent executes an action if it is confident that it is able to draw well from experience. Otherwise, the agent requests for expert help. In the latter, the teacher can *voluntarily* improve the policy learned by the agent through a demonstration.

Also very similar to this work on interactive policy learning is the dogged learning framework for robots Grollman and Jenkins [2007] which also provides a protocol for teacher intervention and selectively executing policy autonomously. It has to be noted that these works pertain only to the **model-free** class of imitation learning algorithms that use a classifier to learn state-action mappings. Thus, the algorithm does not utilize the complete power of generalization in the form of rewards. The only kind of generalization that occurs is by drawing information from state-action pairs that have very similar features.

As part of the confidence execution algorithm (refer Algorithm 5), two different situations force the learner to ask for help. First, the learner could have reached a completely *unfamiliar* state. Second, the learner could have reached a state where its own knowledge is *ambiguous*. Chernova and Veloso [2009] also provide heuristics that help evaluate this ambiguity and unfamiliarity. These techniques involve examining the distance and labels of nearest neighbors of the point . The heuristics provided maintain a parameter called τ_{dist} and $\tau conf$ for determining when a state is highly unfamiliar and when there is little confidence in the state's label due to overlap of labels.

The corrective demonstration component of the algorithm is provided because of the concern that the classifier might *over-generalize* and *over-confidently* execute. Thus, mistakes are bound to happen which need to be corrected. They authors therefore provide a framework wherein the teacher can intervene and correct these mistakes. While here the **onus is on the teacher to correct mistakes**, what we aim in our work is a framework where the learner is more self-aware in that it never makes mistakes and hence the teacher no longer has to intervene voluntarily. If at all the does go wrong, we would like to make sure that these mistakes are bounded. However, we have not explored this relaxation yet.

Algorithm 5 Confidence-Based Autonomy algorithm: Confident Execution and Corrective Demonstration

```
Require: Environment, Teacher, Learner, Classifier $\mathcal{C}$
  $\tau_{conf} \leftarrow \infty$
  $\tau_{dist} \leftarrow 0$
  while True do
      $s \leftarrow$ Current State
      $d \leftarrow$ Distance threshold of $s$
      $c \leftarrow$ Confidence threshold of $s$
      $a \leftarrow \mathcal{C}(s)$
      if $c > \tau_{conf}$ and $d < \tau_{dist}$ then
          Execute $a$
      else
          $a \leftarrow$ Demonstration from Teacher
          Update $\mathcal{C}$ with $(s, a)$
          Update $\tau_{dist}, \tau_{conf}$
          Execute $a$
      end if
  end while
```

3.3.2 Active Inverse Reinforcement Learning

3.3.2.1 Active LfD for Robust Autonomous Navigation

Silver *et al.* [2012] study the problem of active learning from demonstrations for the special case of navigation where we have a notion of a start and goal states. The authors study both model-free and model-based approaches to this problem. In the model-based approach they consider the setup where each state-action pair is assigned a ***cost*** that is parametrized. What is assumed here is that these cost parametrizations somehow encapsulate the structure of the MDP. Recall that this cost parametrization is essentially $\mathbf{\Phi}_Q$.

The way this work looks at two different factors that have to be considered for querying on a state. These factors which they call as *novelty* and *uncertainty* are equivalent to *unfamiliar* and *ambiguous* knowledge that are discussed in Chernova and Veloso [2009]. To query for knowledge that is novel, the authors propose that we must ask the expert for a demonstration from a start state to a goal state that are chosen in such a way that the path between these two traverses through unseen states when tread according to the current cost hypothesis. Every time a query is made, the cost hypothesis is understood to be updated.

To query for reducing uncertainty in the knowledge, we assume that the uncertainty can be first of all modelled. Once it is modelled, we could ask for demonstrations between start and goal states chosen such that the plan according to the current cost hypothesis passes through uncertain states. Alternatively, we could choose these states in such a way that the plan under the current hypothesis is the most variant under the uncertainty in the costs.

While the authors provide interesting heuristics to tackle the problem of active learning for cost-based approaches, the work lacks explicit guarantees specific to a given MDP problem - which is what we aim at in our work.

3.3.2.2 Active Learning in Bayesian Inverse Reinforcement Learning

In contrast to most other work in active learning for learning from demonstrations, Lopes *et al.* [2009] explicitly deal with inverse reinforcement learning; the authors propose heuristics to make queries to the expert for demonstrations on arbitrary state and update its inference about the rewards.

Their idea is anchored on bayesian IRL algorithms. We have seen in Section 3.2.4 that Bayesian IRL algorithms provide posterior distributions over the reward functions based on the set of demonstrations. Lopes *et al.* [2009] use this effectively to evaluate the need for querying on a state. We might be tempted to state that we must query on the state where the distribution over the rewards is not 'concentrated' (has high entropy). However, this is not a strong notion as it is the learned policy that eventually matters to us from a given reward setting; two very different reward setting could still lead to the same policy. Thus, the authors state that the state with the most entropy over the action to be chosen with respect to the posterior reward distribution is the state that has to be queried on.

Algorithm 6 Active Inverse Reinforcement Learning Protocol

```
Require: Prior demonstration $\mathcal{D}$
  while True do
    Estimate posterior $Pr(\mathbf{R}|\mathcal{D})$
    for $s$ in $\mathcal{S}$ do
      Compute $\hat{H}(s)$, the entropy over the policy at $s$
    end for
    Query expert for action at $s^* = \arg\max_s \hat{H}(s)$
    Expert returns $a^*$
    $\mathcal{D} = \mathcal{D} \cup \{(s^*, a^*)\}$
  end while
```

Our work however differs on various aspects from the above protocol. First, we do not make the assumption that the learner can query on any state. Many of the states may not be realizable at all. We make a stricter assumption that we can query the expert only on the state that we are currently in.

Next, we provide a lot of autonomy to the learner in that the learner can decide to execute an action on its own. However, the work that we described above makes many offline queries which are not thoroughly guaranteed to be *useful* queries. The usefulness of a query can be evaluated in two ways. First, the learner might end up querying on states where the entropy over the policy at a state underestimates the confidence of the current policy on the state. Thus, though we actually know what to do, we think we do not know what to do, and hence make a query! This is the prime factor that KWIK provides us.

Secondly, even if the high entropy at a state matches a low level of confidence at the state, learning a demonstration at the state may not help us learn anything much for the remaining part of the state space. The state we queried on might be an unrealizable state that is far different from all the other states that we are not able to generalize further. These issues have not been considered in their work.

3.3.3 Generalization in Apprenticeship Learning

In this section we summarize how various works have approached the problem of generalization in imitation learning differently. We have already seen that Chernova and Veloso [2009] and Chernova and Veloso [2007] use ideas from classification algorithms that take care of generalizing and also evaluating the generalization for model-free algorithms. Judah *et al.* [2012*a*] use the active PAC-learner routine which abstracts away the process of generalization and in turn provides $(\epsilon, \delta)-$guarantees.

We must note that the generalization inherent in the model-free approach to IRL is poorer when compared to that inherent to IRL algorithms because the MDP structure is made use of in IRL algorithms. However, IRL methods do not explicitly generalize - the parametrization encapsulates the process of generalization across unseen states neatly. However, it is essential to understand this better in order to give theoretical guarantees for these algorithms. It is this lack of understanding that prevents us from bounding the number of queries that the active learner asks in Lopes *et al.* [2009].

Boularias and Chaib-draa [2010] consider the IRL problem under the constraint that the trajectories provided span a small part of the subspace. The authors first assume that the learner constructs a partial policy covering a small part of the MDP seen by the expert trajectories. Next, this policy is somehow generalized across the whole state space. While this can be the final policy that we want, we could also use this policy to generate more trajectories and perform an IRL algorithm.

In order to *transfer* the expert's policy (that covers the demonstrated space) to the undemonstrated space, the authors use a technique called as *soft homomorphism* Sorg and Singh [2009]. Homomorphism between two MDPs is a way of modeling equivalences between the states of the MDPs. This formulation helps us to transfer a policy on one MDP to an equivalent policy in another MDP. However, conventionally homomorphism between two MDPs is computed using our knowledge about both the transition functions and the reward functions of the MDP. The authors here ignore the constraint on the reward functions to study the homomorphism between the MDP corresponding to the demonstrated space and the whole MDP.

Melo and Lopes [2010] consider a relatively simpler approach to generalization for model-free imitation learning that is based on kernels corresponding to a state space metric. (An example function for the kernel would be the bisimulation metric studied by Ferns *et al.* [2012].) The intuition behind the authors' algorithm is that experience for an unseen state is drawn from all states but is weighted according to the distance of the state. Based on this, the authors also suggest an active learning algorithm that queries on states based on the variance in the estimate of the generalized parameters.

3.3.4 Generalizing Apprenticeship Learning across Hypothesis Classes

It is necessary to cite the work of Walsh *et al.* [2010] which might be confused to be overlapping with our contribution to KWIK-learning for inverse reinforcement learning.

The authors in this work design a protocol for a learning agent to be assisted by the teacher. The teacher also provides $\epsilon-$optimal traces to guide the learner. They define formally an appropriate model for apprenticeship learning model and link it to KWIK and Mistake Bound.

The work however is very different from our goal as they do not deal with an inverse reinforcement learning agent. The agent has access to the rewards themselves and instead is enriched with information about the teacher's policy. Despite access to rewards, the authors call it apprenticeship learning possibly because the learner gains demonstrations from the teacher.

In fact the authors themselves state explicitly in their work that their scenario is different from inverse reinforcement learning studied by Abbeel and Ng [2004] where the reward function is deduced from trajectories. However, the authors here deal with an agent which can observe the actual rewards and transitions induced by the teacher's policy and learns to maximize this known reward.

CHAPTER 4

KWIK Inverse Reinforcement Learning

4.1 Motivation

We saw in Section 1.2 that imitation learning is broadly solved in two different ways. One approach is to pose it as a supervised learning problem where a classifier learns the action labels for all states in the state space based on training data - the expert's demonstrations. The other approach is a model-based solution that uses inverse reinforcement learning (IRL) to find a mapping from states to real-valued rewards that makes the expert trajectories seem optimal. The learner hence follows the optimal policy on these rewards. IRL methods have the advantage of representing the acquired knowledge succinctly as rewards over the states.

In practice, both these approaches could suffer from a considerable **burden on the teacher** who is expected to produce sufficient trajectories for accurate imitation. What makes this more undesirable is that many of the trajectories happen to be redundant and yet expensive. To address this, it wouldbe of interest to study active learning approaches to imitation learning. In active learning, the learner actively chooses points from a pool of unlabelled data, and requests the expert for the label.

In model-free imitation learning techniques, the states from the pool of points and the expert action at that states would be the label. Judah *et al.* [2012*b*] (Section 3.3.1) have proposed active learning algorithms for supervised learning based imitation learning and analyse the performance of these algorithms with respect to the performance guarantees provided by the active learning algorithm used for the classification.

On the other hand, there has been very little work on *formally* understanding active imitation learning through IRL. Silver *et al.* [2012] (Section 3.3.2.1) have studied active learning heuristics where the learner requests trajectories in such a way that the knowledge about the reward function acquired is either novel or reduces uncertainty in the current beliefs. Melo and Lopes [2010] (Section 3.3.2.2) use the power of Bayesian IRL (Ramachandran and Amir [2007], Section 3.2.4) to provide an empirical technique to actively query by choosing states. The state that is chosen to be queried is one which has the greatest entropy in the posterior distribution over the policy which is in turn derived from a posterior distribution over the rewards as computed by the Bayesian IRL algorithm.

A drawback with all of the above active learning algorithms is that they assume that the learner has complete access to the state space and can also query the expert for a demonstration on any of these states. Often this might not be desirable because some states may not even be realizable and furthermore, this knowledge might not be accessible to the learner. Judah *et al.* [2011] (Section 3.3.1.1) dwell on this problem and propose that the teacher must inform the learner as to whether its query is out of scope. This will however still impose sufficient burden on the teacher. Chernova and Veloso [2009] and Chernova and Veloso [2007] address this by allowing the learner to *interactively* (Section 3.3.1.2) request the teacher's demonstrations whenever it encounters a state where its confidence on the learned policy is below a threshold. They provide an algorithm that pertains only to the model-free imitation learning and also present empirical results. It is however not clear as to what would be the best parameter to choose for the confidence thresholds that used by this algorithm.

4.2 Contribution

IRL in the KWIK framework To overcome these multiple issues, we propose the novel idea of considering IRL via imitation learning in the Knows-What-It-Knows (KWIK) framework of Li *et al.* [2011]. A KWIK algorithm (Refer Section 2.5) is an online learning algorithm that is considered to be self-aware i.e., if and only if the learner believes that it has insufficient experience to predict on a new sample, does the learner ask the expert for the answer.

Considering imitation learning in this framework significantly benefits us in many ways.

First and most importantly, in the KWIK framework, the burden on the expert is substantially reduced as the learner only selectively requests demonstrations.

While the above advantage can be enjoyed in any active learning framework, we also overcome the problems that come with allowing the learner to query on any arbitrary state. The learner is now forced to make queries on only those states that it visits while learning to imitate. This also ensures that the teacher need not worry about knowing whether a state is realizable or not (which is the case considered by Judah *et al.* [2011], Section 3.3.1.1).

While being active, we are now also able to allow the learner to enact its policy and learn on-the-fly *interactively*. This is different from having an active learner that chooses points with no restriction whatsoever from a pool of unlabelled points.

The KWIK framework allows us to formally evaluate the algorithms in terms of their performance (how likely is the algorithm going to perform sufficiently well?) and in terms of the sample complexity (how many queries will be made to the teacher?). This perspective is lacking in all of the active learning works for IRL and any of the interactive policy learning mechanisms that have been studied for both approaches to imitation learning.

In the KWIK framework where the learner is *self-aware*, we are guaranteed that the learner does not mistakenly assume that it knows what to do when it actually does not! This is of practical importance because we would not want the learner to take a non-optimal and possibly dangerous action which could have been avoided if the expert had intervened.

We study model-based but not model-free imitation learning in the KWIK framework because model-based learning helps us translate the guarantees provided by KWIK to guarantees for the optimality of the learner's policy i.e., how good the *value* of the policy is. Model-free learning is not cost-based. The KWIK IRL algorithm that we formulate ensures that the learned policy is $\epsilon-$optimal.

Though Walsh *et al.* [2010] (Section 3.3.4) study what is called as a generalized apprenticeship learning protocol in relation to KWIK learnable classes, their problem domain, as they claim, is fundamentally different from the imitation learning problem that we consider. They study a learner that has access to the rewards during exploration, while the teacher augments this knowledge.

KWIK IRL to KWIK Classification Next, we propose a reduction of the KWIK apprenticeship learning problem via IRL to KWIK classification. Note that this reduction to classification is not the same as direct imitation learning methods that use a classifier to learn a mapping from states to actions. We are primarily interested in finding the unknown reward function defined over the state-action pairs and not just what action label a state corresponds to. That is, we have a notion of learning *cost* parameters which is absent in the model-free methods. *We show that learning the reward function is equivalent to learning the separating hyperplane in the classification problem.*

Defining KWIK Binary Classification Our next contribution in this work is a novel definition of KWIK classification that applies to the above equivalence in imitation learning. The KWIK framework requires that the learner achieves point-wise accuracy, unlike in a PAC-learner. That is, if the learner makes a prediction on a new sample without seeking expert advice, the learner must be ϵ-accurate. However, it is not possible to define ϵ-accuracy on discrete labels (unless we consider a continuous action space in which case we would opt for KWIK online regression algorithms Strehl and Littman [2007].

One could overcome this by defining accuracy with respect to the prediction about the distance of the sample from the separating hyperplane, as considered in a *selective sampling* algorithm studied by Cesa-Bianchi and Orabona [2009], **?** (Section 3.1.6). However, these algorithms have significantly different assumptions than that expected by the KWIK imitation learning agent and are hence not applicable.

We further motivate the validity of our definition of KWIK classification by providing polynomial KWIK bounds for 1-D classification. Finally, we also provide a KWIK protocol for imitation learning that uses an underlying KWIK classifier that suits our requirement that the learner takes ϵ-optimal actions.

We also discuss ideas for a possible mutli-dimensional classifier.

4.3 KWIK-Learner for Binary Classification

Definition 14 *We define* ***an admissible KWIK-learner for binary classification*** *as follows. We assume that the hypothesis class is the set of separating hyperplanes* $\{\boldsymbol{\theta} | \boldsymbol{\theta} \in \mathbb{R}^k, ||\boldsymbol{\theta}||_2 = 1\}$ *that pass through the origin. The input space is defined to be* $\{\mathbf{x} | \mathbf{x} \in \mathbb{R}^k, ||\boldsymbol{\theta}||_2 = 1\}$*. If the adversary picks a* $\boldsymbol{\theta}_E$*, the correct label of* $\mathbf{x} \in \mathbb{R}^k$ *is given by* $SGN[\boldsymbol{\theta}_E \cdot x] \in \{+1, -1\}$*. For the learner to be admissible, the following must hold good for every run, with probability* $1 - \delta$ *:*

- *Whenever the algorithm makes a prediction on* x*, if* $|\boldsymbol{\theta}_E \cdot \mathbf{x}| > \epsilon$*, then* $\hat{h}(\mathbf{x}) = SGN[\boldsymbol{\theta}_E \cdot \mathbf{x}]$
- *The number of time-steps for which the algorithm emits* $\perp$ *is bounded by* $B(\epsilon, \delta)$ *a function that is polynomial in* $1/\epsilon$*,* $1/\delta$ *and* k*.*

Intuitively, we require that the algorithm predicts correctly on all the points that are sufficiently far away from the separating hyperplane. However, if the sample point is within the $\epsilon-$margin of the hyperplane, we allow the classifier to make mistakes. Furthermore, the classifier can make unbounded number of mistakes within this region.

Visualizing the problem It is important to understand the specifications of the algorithm precisely. We first of all constrain the input points to the surface of a hypersphere of radius 1 unit in $\mathbb{R}^k$ and centered in the origin. The separating hyperplane is a plane in this space that passes through the origin of the sphere and slices the hypersphere surface into two halves each corresponding to one of the labels, $+1$ or -1. $\boldsymbol{\theta}_E$ corresponds to the normal of the hyperplane. The value $|\boldsymbol{\theta}_E \cdot \mathbf{x}| = ||\boldsymbol{\theta}_E||_2||\mathbf{x}||_2 \cos \boldsymbol{\Phi} = 1 \cdot 1 \cdot \cos \boldsymbol{\Phi}$ corresponds to the cosine of the angle $\boldsymbol{\Phi}$ between the normal of the hyperplane and the point $\mathbf{x}$ itself. When this value is low, it means that the point lies orthogonal to the normal and is hence very close to the hypersphere itself. Note that we could allow the inputs to be anywhere in $\mathbb{R}^k$ but still project the point onto the unit hypersphere. Or, we could appropriately modify the accuracy condition to be the following:

$$|\boldsymbol{\theta}_E \cdot \mathbf{x}| \leq ||\mathbf{x}||_2 \epsilon \tag{4.1}$$

This may be compared to the KWIK-MB model proposed by Sayedi *et al.* [2010] where the KWIK algorithm is also allowed to make a fixed number of mistakes. However, our model is significantly different in that we allow infinitely many mistakes but restrict them to a very small space around the separating hyperplane. If we did not allow the learner to make infinitely many mistakes, we would expect the learner to perennially refine its knowledge in the small space around the hyperplane. This might require exponentially many queries to accurately place the hyperplane. Furthermore, we will see that our condition also eventually suits our imitation learning problem where the learner is required to be ϵ-optimal.

Next, we discuss assumptions about noise in the expert's labels. In the KWIK framework, we assume that the noisy observation produced by the expert has an expectation equal to the correct output. Thus, for classification we assume a teacher to be ϵ_Y-optimal, if the teacher outputs y for an input x such that:

$$\begin{aligned} \mathbb{E}[y] &> \epsilon_Y && \text{if } \boldsymbol{\theta}_E \cdot x \geq 0 \\ \mathbb{E}[y] &< -\epsilon_Y && \text{if } \boldsymbol{\theta}_E \cdot x < 0 \end{aligned}$$

In other words, we expect that for any input point, the expert labels it correctly with probability at least $1/2 + \epsilon_Y/2$. A good teacher will ofcourse have a high ϵ_Y. We note that this is a possibly relaxed assumption when compared to the selective sampling approach of Cesa-Bianchi and Orabona [2009] where they assume that the accuracy of the expert increases with the distance from the separating hyperplane. However, in regions close to the hyperplane the teacher is noisier in the latter.

4.3.1 A simple KWIK 1-D classification algorithm

We analyse a naive algorithm for 1-D classification to demonstrate how the KWIK conditions we proposed allow us to design a lgorithms with a KWIK-bound polynomial in $1/\epsilon$ and $1/\delta$ even under the relaxed noise assumption of the teacher's outputs. Here ϵ is the accuracy requirement of the KWIK algorithm; ϵ_Y is the noise parameter of the teacher.

Assume the input space spans unit length. The learning algorithm discretizes the input space into $\frac{1}{\epsilon}$ segments as shown in Figure 4.1. When the adversary presents a sample belonging to a segment, the algorithm emits $\perp$ when the number of samples already queried in this segment is fewer than $\mathcal{O}(\frac{1}{\epsilon_Y^2}\ln(\frac{2}{\epsilon_Y\delta}))$ (these are the segments marked with a question mark in Figure 4.1). When the number of samples is however greater than this we can show that if the segment does not contain the separating point, the proportion of queried points in this segment that would have been labeled correctly by the expert will be at least $1/2 + \epsilon_Y/2 - \epsilon_Y > 1/2$ with a high probability of $1 - \epsilon_Y\delta/2$ (Refer Lemma 5 in Appendix). Thus, after acquiring sufficient samples in each of the segments, we will correctly learn the labels of all the segments *outside an $\epsilon-$ margin* of the separating point with probability at least $1 - \delta$, which is what we want. Thus, the number of queries made will be $\mathcal{O}(\frac{1}{\epsilon_Y\epsilon^2}\ln(\frac{1}{\epsilon_Y\delta}))$.

Remark 1 *The above example is not truly consistent with the KWIK classifier conditions as described in Definition 14. In the definition, we enforced that the separating plane must pass through the origin. Hence for the 1D case, there is only one allowed separating point* *which is nothing but the origin itself. We however do not make this assumption just to illustrate the situation.*

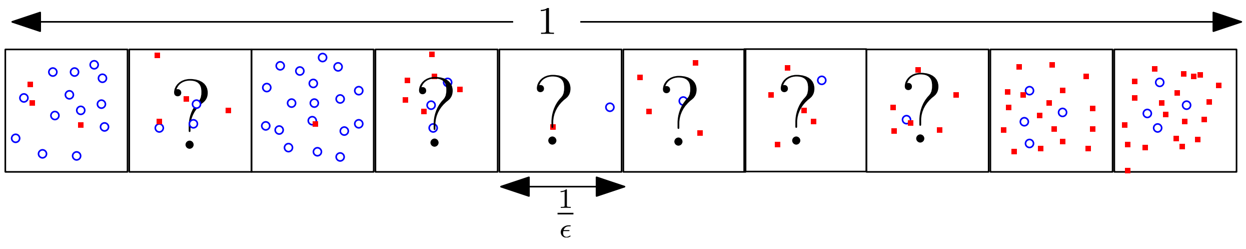

Figure 4.1: Visualization of the naive 1D classification

4.4 KWIK Inverse Reinforcement Learning Protocol

We now present the reduction of IRL to KWIK classification.

Assumption 1 *We assume that the MDP is specified using the* $\mathbf{\Phi}_Q$ *parametrization as discussed in Section 2.3.*

While often it is the case that the problem is parametrized in terms of $\mathbf{\Phi}_R$, that is the rewards, here we follow the approach of Silver *et al.* [2012] in *directly* assigning long-term cost parametrization to the decisions taken. Note that this by default encodes the MDP structure. We might however be interested in beginning with the $\mathbf{\Phi}_R$ parametrization which we will leave for future work.

Assumption 2 $\forall (s,a) \in \mathcal{S} \times \mathcal{A}$, $||\mathbf{\Phi}_Q(s,a)||_2 \leq 1$

The above assumption is valid because any parametrization that we begin with can be made to satisfy this by a simple scaling *without changing the task at hand.*

At any state $s \in \mathcal{S}$, the learner is presented with a set of at most $|\mathcal{A}|$ actions of which the learner is required to pick an $\epsilon-$optimal action. Let $\boldsymbol{\theta}_E$ be the unknown weight vector for the rewards. We assume that the learner has access to a KWIK classification algorithm $\mathcal{C}$ whose input space is $\mathbb{R}^k$. For some $a^*, a' \in \mathcal{A}$, we expect the classifier to predict $\text{SGN}[\boldsymbol{\theta}_E \cdot (\boldsymbol{\Phi}_Q(s,a^*) - \boldsymbol{\Phi}_Q(s,a'))]$ given $(\boldsymbol{\Phi}_Q(s,a^*) - \boldsymbol{\Phi}_Q(s,a'))$ as input.

Assumption 3 *We will assume that the input to the classifier is always scaled to a magnitude of 1.*

The above assumption does not change our problem because scaling $\boldsymbol{\Phi}_Q(s,a^*) - \boldsymbol{\Phi}_Q(s,a')$ to

$$\frac{\boldsymbol{\Phi}_Q(s,a^*) - \boldsymbol{\Phi}_Q(s,a')}{||\boldsymbol{\Phi}_Q(s,a^*) - \boldsymbol{\Phi}_Q(s,a')||_2}$$

does not change the sign of $\text{SGN}[\boldsymbol{\theta}_E \cdot (\boldsymbol{\Phi}_Q(s,a^*) - \boldsymbol{\Phi}_Q(s,a'))]$. That is,

$$\text{SGN}\left[\frac{\boldsymbol{\Phi}_Q(s,a^*) - \boldsymbol{\Phi}_Q(s,a')}{||\boldsymbol{\Phi}_Q(s,a^*) - \boldsymbol{\Phi}_Q(s,a')||_2}\right] = \text{SGN}\left[\boldsymbol{\Phi}_Q(s,a^*) - \boldsymbol{\Phi}_Q(s,a')\right]$$

Thus, in the rest of the discussion we assume that the input to the classifier is scaled appropriately without explicitly mentioning it.

Lemma 1 $\forall (s,a^*), (s,a') \in \mathcal{S} \times \mathcal{A}$

$$||\boldsymbol{\Phi}_Q(s,a^*) - \boldsymbol{\Phi}_Q(s,a')||_2 \leq 2 \tag{4.2}$$

Theorem 2 *If the accuracy parameter of $\mathcal{C}$ is set at*

$$\frac{\epsilon}{2(|\mathcal{A}| - 1)}$$

then:

for the input $\mathbf{\Phi}_Q(s,a^*) - \mathbf{\Phi}_Q(s,a')$, *if*

$$\boldsymbol{\theta}_E \cdot (\mathbf{\Phi}_Q(s,a^*) - \mathbf{\Phi}_Q(s,a')) \geq \frac{\epsilon}{|\mathcal{A}|-1} \tag{4.3}$$

then $\mathcal{C}$ *predicts* $+1$ *if it makes a valid prediction;*

and if

$$\boldsymbol{\theta}_E \cdot (\mathbf{\Phi}_Q(s,a') - \mathbf{\Phi}_Q(s,a^*)) \geq \frac{\epsilon}{|\mathcal{A}|-1} \tag{4.4}$$

then $\mathcal{C}$ *predicts* -1 *if it makes a valid prediction.*

Proof If

$$\boldsymbol{\theta}_E \cdot (\mathbf{\Phi}_Q(s,a^*) - \mathbf{\Phi}_Q(s,a')) \geq \frac{\epsilon}{|\mathcal{A}|-1} \tag{4.5}$$

then

$$\begin{aligned}\boldsymbol{\theta}_E \cdot \frac{(\mathbf{\Phi}_Q(s,a^*) - \mathbf{\Phi}_Q(s,a'))}{||\mathbf{\Phi}_Q(s,a^*) - \mathbf{\Phi}_Q(s,a')||_2} &> \frac{\epsilon}{|\mathcal{A}-1|} \times \frac{1}{||\mathbf{\Phi}_Q(s,a^*) - \mathbf{\Phi}_Q(s,a')||_2} \\ &\geq \frac{\epsilon}{|\mathcal{A}-1|} \times \frac{1}{2}\end{aligned}$$

The first inequality here comes from the assumption we made. The second inequality comes from Lemma 1. Thus, the dot product between the normalized input and the true weight vector is strictly greater than $\frac{\epsilon}{2(|\mathcal{A}-1|)}$ which is the accuracy parameter of the classifier. Hence, we know that the KWIK classification algorithm, if at all made a valid prediction, had to make the right prediction, which is $+1$.

We skip the proof for the second result as it is identical. ■

Theorem 3 *If the accuracy parameter of $\mathcal{C}$ is set at*

$$\frac{\epsilon}{2(|\mathcal{A}|-1)}$$

then:

for the input $\mathbf{\Phi}_Q(s,a^*) - \mathbf{\Phi}_Q(s,a')$, if $\mathcal{C}$ predicts $+1$, then

$$\boldsymbol{\theta}_E \cdot (\mathbf{\Phi}_Q(s,a^*) - \mathbf{\Phi}_Q(s,a')) \geq -\frac{\epsilon}{|\mathcal{A}|-1} \tag{4.6}$$

If $\mathcal{C}$ predicts -1, then

$$\boldsymbol{\theta}_E \cdot (\mathbf{\Phi}_Q(s,a') - \mathbf{\Phi}_Q(s,a^*)) \geq -\frac{\epsilon}{|\mathcal{A}|-1} \tag{4.7}$$

Proof The proof for the above results are identical. Hence, we prove the first result. Let the prediction made by the KWIK classifier be $+1$. Assume on the contrary that

$$\boldsymbol{\theta}_E \cdot (\mathbf{\Phi}_Q(s,a^*) - \mathbf{\Phi}_Q(s,a')) < -\frac{\epsilon}{|\mathcal{A}|-1}$$

Then, from Theorem 2, we know that the classifier has to have output -1. Since this contradicts the fact that $\mathcal{C}$ predicted $+1$, we prove that our assumption is wrong. ■

Thus, whenever the classifier makes a valid prediction and favors one action over the other, we see that the preferred action cannot be worse than the other action by a value of more than $\epsilon/(|\mathcal{A}|-1)$.

If the classifier is unable to predict, and instead outputs a $\perp$ we request expert advice in the form of a preference over these pair of actions. We note that we could study various other modifications of this algorithm, where the expert only provides knowledge about the best action amongst all actions instead of pairwise preferences.

We present below the protocol followed by the learner in order to choose the $\epsilon-$optimal action if it takes a decision autonomously.

Algorithm 7 KWIK Inverse Reinforcement Learning Protocol

```
Require: Teacher $\mathcal{T}(\epsilon_Y)$ with true weight vector for rewards $\boldsymbol{\theta}_E$, Admissible KWIK
  Classifier $\mathcal{C}(\frac{\epsilon}{2(|\mathcal{A}|-1)}, \delta)$ with weight estimate for rewards $\hat{\boldsymbol{\theta}}$
  for $t = 1, 2, \ldots$ do
      $s$ = Current State of the Environment
      $\hat{a}_{best} = a_1$
      for i $= 2, \ldots |\mathcal{A}|$ do
          Present $\mathbf{\Phi}_Q(s, a_i) - \mathbf{\Phi}_Q(s, \hat{a}_{best})$ to $\mathcal{C}$
          if Output of $\mathcal{C} = \perp$ then
              Present $\mathbf{\Phi}_Q(s, a_i) - \mathbf{\Phi}_Q(s, \hat{a}_{best})$ to $\mathcal{T}$
              $\mathcal{T}$ outputs SGN$[\boldsymbol{\theta}_E \cdot (\mathbf{\Phi}_Q(s, a_i) - \mathbf{\Phi}_Q(s, \hat{a}_{best}))]$
              $\mathcal{C}$ learns from output of $\mathcal{T}$ and updates $\hat{\boldsymbol{\theta}}$
              if Output of $\mathcal{T} = +1$ then
                  $\hat{a}_{best} = a_i$
              end if
          else
              $\mathcal{C}$ outputs SGN$[\hat{\boldsymbol{\theta}}_E \cdot (\mathbf{\Phi}_Q(s, a_i) - \mathbf{\Phi}_Q(s, \hat{a}_{best}))]$
              if Output of $\mathcal{C} = +1$ then
                  $\hat{a}_{best} = a_i$
              end if
          end if
      end for
      Perform $\hat{a}_{best}$
  end for
```

In Algorithm 7, at any state the learner scans all the possible actions (which is at most $|\mathcal{A}|$) and maintains a candidate action that it considers to be the best amongst the actions that have been iterated through. The correctness of this algorithm would follow if we show that after iterating over all the actions, if the algorithm has not queried the teacher, it always chooses an ϵ-optimal action. We prove the following lemma from which the above statement follows directly by setting $i = |\mathcal{A}|$.

Theorem 4 *After iterating over the first i actions, if the algorithm has not made any queries, the candidate action picked by the algorithm is $(i-1)\frac{\epsilon}{|\mathcal{A}|-1}-$ optimal with respect to the best action amongst the first i actions.*

Proof The claim trivially holds good when $i = 1$. For any arbitrary round $i < |\mathcal{A}|$ assume that the claim is true. This is our inductions hypothesis. That is, if $\hat{a}_i$ is the candidate action picked by the algorithm, and a_i^* is the best action amongst the first i actions, then:

$$\boldsymbol{\theta}_E \cdot \boldsymbol{\Phi}_Q(s, \hat{a}_i) \geq \boldsymbol{\theta}_E \cdot \boldsymbol{\Phi}_Q(s, a_i^*) - (i-1)\frac{\epsilon}{|\mathcal{A}|-1} \tag{4.8}$$

If the algorithm chose a_{i+1} over $\hat{a}_i$, we know from Theorem 3 that

$$\boldsymbol{\theta}_E \cdot \boldsymbol{\Phi}_Q(s, a_{i+1}) \geq \boldsymbol{\theta}_E \cdot \boldsymbol{\Phi}_Q(s, \hat{a}_i) - \frac{\epsilon}{|\mathcal{A}|-1} \tag{4.9}$$

If the decision of choosing a_{i+1} over $\hat{a}_i$ were to be inconsistent with our claim, a_{i+1} must not be an $i\frac{\epsilon}{|\mathcal{A}|-1}$-optimal action (among the first $i+1$ actions). Then a^* must *still* be the best action amongst the first $i+1$ actions. However, from inequalities (4.8) and (4.9), we can see that:

$$\boldsymbol{\theta}_E \cdot \boldsymbol{\Phi}_Q(s, a_{i+1}) \geq \boldsymbol{\theta}_E \cdot \boldsymbol{\Phi}_Q(s, a^*) - i\frac{\epsilon}{|\mathcal{A}|-1}$$

which makes a_{i+1} an $i\frac{\epsilon}{|\mathcal{A}|-1}$-optimal action amongst the first $i+1$ actions, which is a contradiction. Thus, if the algorithm chose a_{i+1} over $\hat{a}_i$, it must have been right.

On the other hand, if the algorithm still chose $\hat{a}_i$ over a_{i+1}, it would be inconsistent with our claim only if a_{i+1} was the best action amongst the $i+1$ actions and if $\hat{a}_i$ was not sufficiently optimal. That is,

$$\boldsymbol{\theta}_E \cdot \mathbf{\Phi}_Q(s, \hat{a}_i) < \boldsymbol{\theta}_E \cdot \mathbf{\Phi}_Q(s, a_{i+1}) - i\frac{\epsilon}{|\mathcal{A}|-1} \tag{4.10}$$

However, this implies a much weaker inequality:

$$\boldsymbol{\theta}_E \cdot \mathbf{\Phi}_Q(s, \hat{a}_i) < \boldsymbol{\theta}_E \cdot \mathbf{\Phi}_Q(s, a_{i+1}) - \frac{\epsilon}{|\mathcal{A}|-1}$$

which would have ensured that the KWIK classifier indicated that a_{i+1} was a better action (Lemma 2). Hence, by induction we show that the lemma holds good for all $i = 1, \ldots |\mathcal{A}|$.

■

Corollary 1 *After scanning over all the actions, if the algorithm has not made any queries, the candidate action picked by the algorithm is an $\epsilon-$ optimal action.*

4.5 Multidimensional KWIK Classifier

We discuss in this section preliminary ideas on how a classifier could be designed for a $k-$dimensional input space.

Let us describe a naive idea first. We follow an idea similar to that of the 1-D classifier discussed in Section 4.3.1. The idea is to split the input space into N appropriate subspaces, and make predictions in these subspaces only after making sufficient number of queries in each of them. We want each segment to fail with atmost δ/N probability so that the total failure probability is atmost δ (Lemma 2). From Lemma 5, each subspace will thus require $\frac{1}{\epsilon_Y^2}\ln\left(\frac{N}{\delta}\right)$ points to ensure that the majority of labels is correct. Thus, we will make as many queries as:

$$N \times \frac{1}{\epsilon_Y^2} \ln\left(\frac{N}{\delta}\right) \tag{4.11}$$

The constraint with splitting the space is that the subspaces must be of width not more than ϵ. We could come up with many different ways of splitting the unit hypersphere surface. We skip discussing these ideas here as they do not form the core of our discussion. However, we must note that such a splitting will always be of of the order $\mathcal{O}(1/\epsilon^k)$. Thus, we will end up making queries of the order:

$$\frac{1}{\epsilon^k \epsilon_Y^2} \ln\left(\frac{1}{\epsilon^k \delta}\right) \tag{4.12}$$

which is ***exponential***!

Our aim in this discussion is to gain insights about how we can *characterize* regions in the input space by generalizing from experience across the space and not just from the local neighborhood. This might *perhaps* lead us to a polynomial bound.

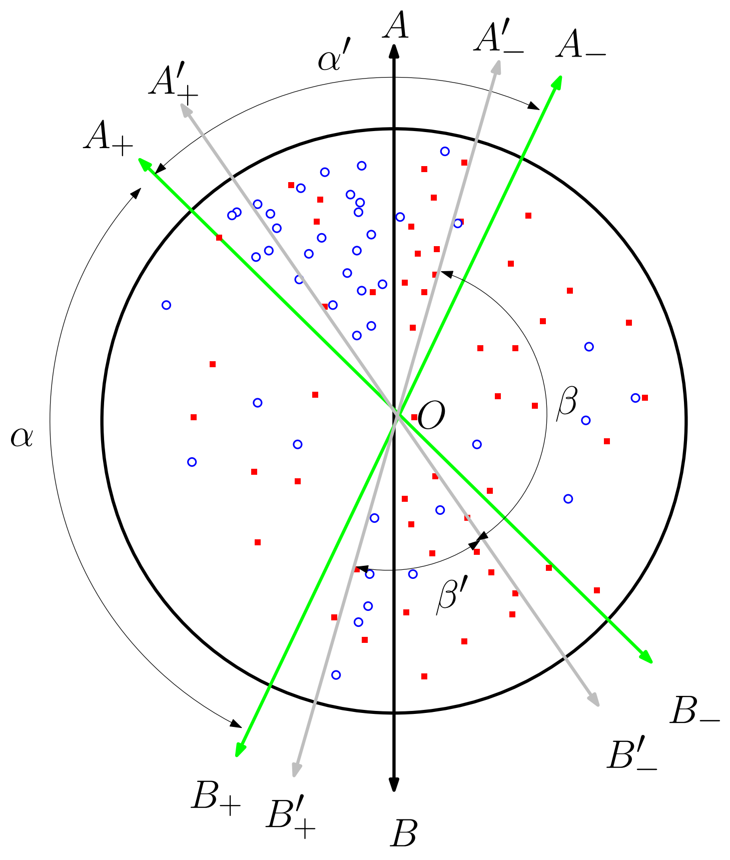

Figure 4.2: Visualization of the k-D classification

Remark 2 *The KWIK classifier imposes stricter restrictions on the learner in imitation learning than that is required of the learner for it to be near-optimal.*

Consider the space of input points which are not normalized. Visually, the subspace where infinitely many mistakes are allowed by the near-optimal learner is a cylindrical margin around the separating hyperplane. The cylinder is of radius $\epsilon/2(|\mathcal{A}-1|)$. However, the classifier $\mathcal{C}$ allows infinitely many mistakes only in a conical region centred at the origin with the base subtended at the unit hypersphere of radius $\epsilon/2(|\mathcal{A}-1|)$. There is space outside of this cone that can still be misclassified for the near-optimality requirements of the learner. This discrepancy comes from the fact that the classifier takes in normalized inputs. However the discrepancy is not invalid in any sense because, the classifier is known to pass through the origin, and any sort of perturbation will be only with respect to its ‘angle’; if the classifier is perturbed the space it sweeps is only a conical region and hence we must allow errors to occur only in this region.

CHAPTER 5

OVERVIEW

In this work we have dealt with understanding the problem of imitation learning in the active learning setting. We have specifically chosen inverse reinforcement learning as a means for solving imitation learning and explored how this can be performed by an active learner.

Furthermore, we have consider this problem in an online *interactive* framework where the learner is allowed to make queries to the expert only on states that it encounters. We have identified the Knows-What-It-Knows framework to be suited for this purpose and also examined the practical relevance of considering such a learner: the burden on the teacher is reduced, the learner knows when to query, and whenever the learner takes a decision autonomously the learner is near-optimal. Besides the practical importance of studying the problem in the KWIK framework, we also see that it helps us provide theoretical guarantees which have not been studied before in the (inter-)active learning algorithms for cost-based imitation learning methods (i.e., inverse reinforcement learning) like Chernova and Veloso [2009] and Silver *et al.* [2012].

To make use of the MDP structure and similarity across state-action pairs we modeled the problem as an MDP with linear parametrization of the costs of decisions. To reduce the problem to a KWIK learning problem, we looked at the queries as queries about comparisons between pairs of parametrized actions. We showed how this reduces to classification, and the problem of finding the weights given to various features of these costs boils down to finding a separating hyperplane passing through the origin.

Our next contribution was to understand classification in the KWIK framework so that the working of the classifier results in the agent taking near optimal actions if it does take an action autonomously. Here, we provided the groundwork an algorithm that works on a multi-dimensional space. We discussed insights that help us understand when the learner can be confident about its prediction and when it *should* not be.

We have thus provided the scaffolding for understanding KWIK Inverse Reinforcement Learning. This work leads us to two different problems that are interesting to us: the problem of understanding KWIK classification and the problem of KWIK IRL itself. We discuss possible future directions in the next section.

CHAPTER 6

FUTURE WORK

There are two primary directions for future work. One of the primary focuses for future work would be to study algorithms for classification in the KWIK framework as defined above. We have provided the groundwork for the classification algorithm which we however understand to have exponential bounds. This could be possible because of the extremely relaxed assumptions on the noise of the teacher. Perhaps with a more restrictive assumption on the noise, that is still practically realizable, we might be able to devise algorithms with polynomial bounds.

The other direction for future work would deal with studying the KWIK IRL problem itself. For example, we would be interested in addressing the issue that the protocol currently asks pairwise queries to determine the near-optimal action. The teacher responds to these queries accordingly. However, ideally we would like the teacher to directly point out the best action. Thus, the algorithm should be adapted to a teacher who merely tells it what the best action is.

We would also like to study the equivalence between the KWIK classification condition and the $\epsilon-$optimal condition. The current equivalence is not strong in that the KWIK classification condition is more restrictive than the $\epsilon-$optimality in imitation learning.

One important question we would like to address is the fact that in the current problem setting we assume that we have access to the cost function for each decision ($\mathbf{\Phi}_Q$) directly.

What would be more interesting is having access only to the short term reward functions ($\mathbf{\Phi}_R$) from which we are required to infer $\mathbf{\Phi}_Q$ by gradually learning more about the policy. Remember that $\mathbf{\Phi}_Q$ is dependent not only on $\mathbf{\Phi}_R$ and the structure of the MDP but also on the policy.

CHAPTER 7

APPENDIX

We list some theorems from probability theory and statistics that have been referred to in our discussion.

Lemma 2 ***Union Bound*** *If events E_i, $i = 1, 2 \ldots m$ have respective probability of occurrences p_i, $i = 1, 2 \ldots m$, then the probability that at least one of the events happens is at most*

$$\sum_{i=1}^{m} p_i \tag{7.1}$$

The probability that none of the events occur is at least

$$1 - \sum_{i=1}^{m} p_i \tag{7.2}$$

Lemma 3 ***Hoeffding Bound*** *If $x_1, x_2, \ldots x_m$ are m independent random variables such that $x_i \in [L_i, U_i]$ and $\mathbb{E}[x_i] = \mu_i$, $\forall i$, and if*

$$\hat{\mu} = \frac{1}{m} \sum_{i=1}^{m} x_i$$

then,

$$Pr\left(\hat{\mu} - \mu \geq \epsilon\right) \leq \exp\left(-\frac{2m^2\epsilon^2}{\sum_{i=1}^{m}(U_i - L_i)^2}\right) \tag{7.3}$$

$$Pr\left(\hat{\mu} - \mu \leq \epsilon\right) \leq \exp\left(-\frac{2m^2\epsilon^2}{\sum_{i=1}^{m}(U_i - L_i)^2}\right) \tag{7.4}$$

We state an implication of the Hoeffding Bound for a specific case which will be of use to us.

Lemma 4 *If $x_1, x_2, \ldots x_m$ are m independent Bernoulli trials such that $\mathbb{E}[x_i] = \mu$ and if*

$$\hat{\mu} = \frac{1}{m}\sum_{i=1}^{m} x_i$$

then,

$$Pr\left(\hat{\mu} - \mu \geq \epsilon\right) \leq \exp\left(-2m\epsilon^2\right) \tag{7.5}$$

$$Pr\left(\hat{\mu} - \mu \leq \epsilon\right) \leq \exp\left(-2m\epsilon^2\right) \tag{7.6}$$

Lemma 5 ***Coin Learning Lemma*** *If $x_1, x_2, \ldots x_m$ are m independent Bernoulli trials such that $\mathbb{E}[x_i] > \epsilon/2 + 1/2$ and if*

$$\hat{\mu} = \frac{1}{m}\sum_{i=1}^{m} x_i$$

then,

$$Pr\left(\hat{\mu} \leq 1/2\right) \leq \exp\left(-m\epsilon^2/2\right) \tag{7.7}$$

Thus, when

$$m = \frac{2}{\epsilon^2}\ln\left(\frac{1}{\delta}\right) \tag{7.8}$$

we can ensure that

$$Pr(\hat{\mu} > 1/2) > 1 - \delta \tag{7.9}$$

LIST OF PAPERS BASED ON THESIS

1. Vaishnavh Nagarajan and Balaraman Ravindran "KWIK Inverse Reinforcement Learning", *The 2nd Multidisciplinary Conference on Reinforcement Learning and Decision Making. (RLDM), 2015* (Accepted).

REFERENCES

Abbeel, P. and **A. Y. Ng**, Apprenticeship learning via inverse reinforcement learning. *In Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04. ACM, New York, NY, USA, 2004. ISBN 1-58113-838-5. URL http://doi.acm.org/10.1145/1015330.1015430.

Auer, P. (2002). Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, **3**, 397–422. URL http://www.jmlr.org/papers/v3/auer02a.html.

Blum, A. L. (1994). Separating distribution-free and mistake-bound learning models over the boolean domain. *SIAM J. Comput.*, **23**(5), 990–1000. ISSN 0097-5397. URL http://dx.doi.org/10.1137/S009753979223455X.

Boularias, A. and **B. Chaib-draa**, Apprenticeship learning via soft local homomorphisms. *In IEEEInternational Conference on Robotics and Automation, ICRA 2010, Anchorage, Alaska, USA, 3-7 May 2010*. 2010. URL http://dx.doi.org/10.1109/ROBOT.2010.5509717.

Cesa-Bianchi, C., Nicolòand Gentile and **F. Orabona**, Robust bounds for classification via selective sampling. *In Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09. ACM, New York, NY, USA, 2009. ISBN 978-1-60558-516-1. URL http://doi.acm.org/10.1145/1553374.1553390.

Chernova, S. and **M. Veloso**, Confidence-based policy learning from demonstration using gaussian mixture models. *In Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS '07. ACM, New York, NY, USA, 2007. ISBN 978-81-904262-7-5. URL http://doi.acm.org/10.1145/1329125.1329407.

Chernova, S. and **M. Veloso** (2009). Interactive policy learning through confidence-based autonomy. *J. Artif. Int. Res.*, **34**(1), 1–25. ISSN 1076-9757. URL http://dl.acm.org/citation.cfm?id=1622716.1622717.

Ferns, N., **P. Panangaden**, and **D. Precup** (2012). Metrics for finite markov decision processes. *CoRR*, **abs/1207.4114**. URL http://arxiv.org/abs/1207.4114.

Grollman, D. H. and **O. C. Jenkins**, Dogged learning for robots. *In 2007 IEEE International Conference on Robotics and Automation, ICRA 2007, 10-14 April 2007, Roma, Italy*. IEEE, 2007. URL http://dx.doi.org/10.1109/ROBOT.2007.363692.

Judah, K., **A. Fern**, and **T. Dietterich**, Active imitation learning via state queries. 2011.

Judah, K., **A. Fern**, and **T. G. Dietterich** (2012*a*). Active imitation learning via reduction to I.I.D.active learning. *CoRR*, **abs/1210.4876**. URL http://arxiv.org/abs/1210.4876.

Judah, K., **A. Fern**, and **T. G. Dietterich**, Active imitation learning via reduction to I.I.D.active learning. *In Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence, Catalina Island, CA, USA, August 14-18,* *2012*. 2012*b*. URL http://uai.sis.pitt.edu/displayArticleDetails.jsp?mmnu=1&smnu=2&article_id=2304&proceeding_id=28.

Li, L., **M. L. Littman**, **T. J. Walsh**, and **A. L. Strehl** (2011). Knows what it knows: A framework for self-aware learning. *Mach. Learn.*, **82**(3), 399–443. ISSN 0885-6125. URL http://dx.doi.org/10.1007/s10994-010-5225-4.

Littlestone, N., Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *In Proceedings of the 28th Annual Symposium on Foundations of Computer Science*, SFCS '87. IEEE Computer Society, Washington, DC, USA, 1987. ISBN 0-8186-0807-2. URL http://dx.doi.org/10.1109/SFCS.1987.37.

Littlestone, N., From on-line to batch learning. *In Proceedings of the Second Annual Workshop on Computational Learning Theory*, COLT '89. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1989. ISBN 1-55860-086-8. URL http://dl.acm.org/citation.cfm?id=93335.93365.

Lopes, M., **F. Melo**, and **L. Montesano**, Active learning for reward estimation in inverse reinforcement learning. *In Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases: Part II*, ECML PKDD '09. Springer-Verlag, Berlin, Heidelberg, 2009. ISBN 978-3-642-04173-0. URL http://dx.doi.org/10.1007/978-3-642-04174-7_3.

Melo, F. S. and **M. Lopes**, Learning from demonstration using MDPinduced metrics. *In Machine Learning and Knowledge Discovery in Databases, European Conference, ECMLPKDD2010, Barcelona, Spain, September 20-24, 2010, Proceedings, Part II*. 2010. URL http://dx.doi.org/10.1007/978-3-642-15883-4_25.

Ng, A. Y. and **S. J. Russell**, Algorithms for inverse reinforcement learning. *In Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2000. ISBN 1-55860-707-2. URL http://dl.acm.org/citation.cfm?id=645529.657801.

Ramachandran, D. and **E. Amir**, Bayesian inverse reinforcement learning. *In Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2007. URL http://dl.acm.org/citation.cfm?id=1625275.1625692.

Ross, S. and **D. Bagnell**, Efficient reductions for imitation learning. *In Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010*. 2010. URL http://www.jmlr.org/proceedings/papers/v9/ross10a.html.

Sayedi, A., **M. Zadimoghaddam**, and **A. Blum**, Trading off mistakes and don't-know predictions. *In Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010. Proceedings of a meeting held 6-9 December 2010, Vancouver, British Columbia, Canada.*. 2010. URL http://papers.nips.cc/paper/4142-trading-off-mistakes-and-dont-know-predictions.

Silver, D., **J. A. Bagnell**, and **A. Stentz**, Active learning from demonstration for robust autonomous navigation. *In IEEEInternational Conference on Robotics and Automation, ICRA 2012, 14-18 May, 2012, St. Paul, Minnesota, USA*. 2012. URL http://dx.doi.org/10.1109/ICRA.2012.6224757.

Sorg, J. and **S. Singh**, Transfer via soft homomorphisms. *In Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 2*, AAMAS '09. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2009. ISBN 978-0-9817381-7-8. URL http://dl.acm.org/citation.cfm?id=1558109.1558114.

Strehl, A. L. and **M. L. Littman**, Online linear regression and its application to model-based reinforcement learning. *In Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007*. 2007. URL http://papers.nips.cc/paper/3197-online-linear-regression-and-its-application-to-model-based-reinforcement-learning.

Syed, U. and **R. E. Schapire**, A reduction from apprenticeship learning to classification. *In Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010. Proceedings of a meeting held 6-9 December 2010, Vancouver, British Columbia, Canada.*. 2010. URL http://papers.nips.cc/paper/4180-a-reduction-from-apprenticeship-learning-to-classification.

Valiant, L. G. (1984). A theory of the learnable. *Commun. ACM*, **27**(11), 1134–1142. ISSN 0001-0782. URL http://doi.acm.org/10.1145/1968.1972.

Walsh, T. J., **K. Subramanian**, **M. L. Littman**, and **C. Diuk**, Generalizing apprenticeship learning across hypothesis classes. *In Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*. 2010. URL http://www.icml2010.org/papers/475.pdf.

Ziebart, B. D., **A. Maas**, **J. A. Bagnell**, and **A. K. Dey**, Maximum entropy inverse reinforcement learning. *In Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3*, AAAI'08. AAAI Press, 2008. ISBN 978-1-57735-368-3. URL http://dl.acm.org/citation.cfm?id=1620270.1620297.