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

A Project Report

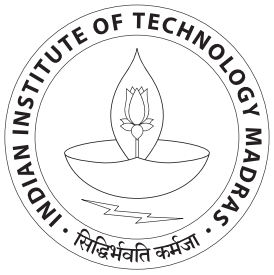
submitted by

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 **Dr. Balaraman Ravindran**, my research adviser, 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.

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

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 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 by 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 such that 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 the study of IRL in the Knows-What-It-Knows (KWIK) framework. We propose a reduction of KWIK IRL to KWIK classification where determining the separating hyperplane becomes equivalent to learning the reward function itself. 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 FIGURES

List of Algorithms

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

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
$\hat{\boldsymbol{\theta}}$	Learner's 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 following expert's policy
$Pr(\cdot)$	Probability of occurrence of an 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
$d_\pi(\cdot)$	Stationary distribution of a policy π
$\mathcal{C}$	KWIK Classification Algorithm
ϵ	Accuracy parameter of a KWIK algorithm
δ	Failure/Confidence parameter of a KWIK algorithm
ϵ_Z	Noise parameter of teacher's outputs

CHAPTER 1

INTRODUCTION

In this section, we will introduce the reader to fundamental concepts in reinforcement learning and machine learning that will be required to motivate and understand the problem we are interested in. The technical preliminaries are postponed to a later section. For the sake of the reader who is unfamiliar with the framework of reinforcement learning, 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 in Section 1.3 we will describe the *inverse* reinforcement learning problem in the context of imitation learning (Section 1.2). Next, in Section 1.4 we will describe the online learning framework of Knows-What-It-Knows (KWIK). Finally, in Section 1.5, we briefly describe the problem that forms the goal of this work.

1.1 Reinforcement Learning

The *full* reinforcement learning (RL) problem is a way of modeling **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** as the agent performs one **action** after the other in each time-step.

The execution of an action results in two kinds of feedback from the environment. First, the agent receives a **reward**. Second, the environment makes a **transition** to another state. Both the above effects are dependent only on the state the agent was in when it took the action, and the action itself - but not any event that happened further back in time. The above system of the agent, the environment, states, actions, transitions and rewards is encapsulated as a **Markov Decision Process**.

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 direction (the action), finds itself in various positions in the maze (the state), and receives a reward (or a punishment) from the maze if it reaches its goal (or hits the wall).

The 'problem' here is that the agent is required to maximize some form of cumulative reward called the **return**. 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 learned. 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 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 noisy 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 behavior 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. The environment is a means of transferring our knowledge to the learner. For example, if we were to make a helicopter robot 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. However, practically, the process of tuning these rewards turns out to be a daunting task for the system designer!

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** as an alternative means for transferring knowledge to the agent. These demonstrations are basically sample trajectories of state

action pairs in the environment as demonstrated by an expert who can enact the behavior that we want. 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 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 'meaninglessly' 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 the return corresponding 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 help the agent discriminate between what sequence of state-action pairs is *good* and what sequence is *bad*. As we saw previously, in inverse reinforcement learning, we no longer have immediate access to such rewards. Instead, we only have trajectories (produced by an expert) that is assumed to be optimal (or close to optimal) with respect to an underlying reward system. The main goal of inverse

reinforcement learning is to understand what these rewards are that the expert's demonstrated behavior is better than any other possible behavior. We could later use these rewards to build our own 'meaningful' policy that is close to that of the expert, which is what imitation learning eventually 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 value that is dependent on various attributes of the state-action pair that it corresponds to. That is, the reward at each state-action pair is encoded as a set of features that describes 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 an appropriately weighted sum of the features of the pair.

In this setup, estimating the reward boils down to determining the weight parameters that translate the state-action pair attributes to the real valued reward. Thus, by estimating these 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.

The generalization that can be done in model-free imitation learning is not as efficient as that in inverse reinforcement learning because of the fact that inferring rewards is a way of understanding the environment and the task more meaningfully. The parameters of this reward function gives us a **succinct representation** of the demonstrated trajec-

tories.

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 analyzing 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 - and these guarantees are a way of evaluating the algorithms. In machine learning, a learner tries to approximate an 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 of the function. 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 receives input points one after the other and dynamically updates its estimate of the function. However, the learner is made to predict the outputs as it receives the points and is evaluated based on these outputs. KWIK simply requires that the algorithm achieves sufficient accuracy *on every individual output*. This might seem like a strong constraint; 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 but chose to predict on.

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

We must also dwell on how the KWIK framework is also associated with **active learning**. In active learning, the learner has the right to choose points from a pool of points for which it would like to know the output. Active learning is relevant when asking for the output on all input points is redundant and adds to the cost of obtaining labels. The learner is encouraged to cleverly choose a subset from the pool of points that is as small in number as possible as it is expensive to ask queries! KWIK happens to incorporate this active form of learning as the learner has the power to abstain from producing a *don't know* when it wants to.

Another significant feature of KWIK algorithms is that they are **self-aware** i.e, a

KWIK algorithm decides to query for the teacher's output on a select set of points and is also aware that it does not need the teacher's output on the remaining points. When the learner makes a prediction and does not ask for the teacher's output, the learner is sufficiently accurate and also knows that it is sufficiently accurate.

We 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, we would have to wisely 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 if the expert demonstrates a lot of trajectories in the same region of the environment. It does not help much 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 enter a state where the learner can *generalize* well from experience seen in possibly many other similar states? It would be redundant to ask for a demonstration in this unseen state. Hence, we would need the power of active learning

which will help us understand when it would be wise to query and when it would not be.

Another question to ask now is the following: assuming we do not make a query to the expert, and allow the learning agent to take an action it thinks is the best from what the agent 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. While this accuracy does not translate to meaningful terms in the model-free imitation learning algorithm, in inverse reinforcement learning based imitation learning, this translates to taking an action that is *almost good* with respect to its return.

1.6 Summary

The rest of the document is organized as follows. Chapter 2 formally introduces the terms and definitions that will be used for discussing the problem. In particular, we will introduce the reader to the notations that are common in reinforcement learning, and also define the Knows-What-It-Knows framework. In Chapter 3, we elaborately survey related work in the literature. This is intended for the reader who is particularly interested in gaining greater technical depth about active inverse reinforcement learning. However, the chapter is *not* a prerequisite for understanding our work. In Chapter 4 we detail our motivation in solving this problem in the light of earlier work and then proceed to discussing our solution and insights. Finally, in Chapter 5 we summarize our contributions following which in Chapter 6 we discuss the various directions for future research.

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 a 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.* $T \in (\mathcal{P}_{\mathcal{S}})^{\mathcal{S}\times\mathcal{A}}$*, 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 note that when* γ *is lower, the cumulative sum gives more weight to the immediate rewards and less weight to the rewards that occur in the distant future.*

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 $s_{t+1} \in \mathcal{S}$ sampled from 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:*

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

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 following 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** $\pi : \mathcal{S} \to \mathcal{A}$ is a mapping from states to actions in the MDP.*

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

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

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

The stationary distribution of π also denotes the following:

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

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 'quality' of a policy.

Definition 5 *The **value function** $V_M^\pi : \mathcal{S} \to \mathbb{R}$ of a policy π is defined as:*

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

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 desirable a state is under a policy, we would also like to see how desirable an action is under a policy. Thus, we will 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:*

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

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

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. Recall that for the problem we deal with we would like the agent to learn a policy that is near-optimal with respect to the underlying rewards on which the teacher's policy is optimal.

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

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

2.2.1 Bellman Equations

While the definitions for the value functions provided a closed form representation, we will now present equations that relate the value functions for one state to the other.

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

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.

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

2.4 Inverse Reinforcement Learning

Definition 9 *In a set of demonstrations $\mathcal{D}$, a **demonstration** $d \in \mathcal{D}$ 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 following a policy in the MDP. That is, the sequence obeys the transition function dictated by the policy and the environment.*

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 determine an estimate of the reward function $\hat{R}$ that optimizes an objective function dependent on $\hat{R}$, $M \backslash R$ and $\mathcal{D}$. The objective function 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 possible reward functions that will explain the demonstrations by the expert that have been seen 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 the literature. One way this can be tackled is by **adding more constraints** to the optimization problem that solves for 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 linearly parametrized 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*

Assumption 1 *For theoretical purposes, we assume that the problem is **realizable** i.e., the target function belongs to the 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.*
- *Now 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 $z_t \in \mathcal{Z}$.*

Remark 1 *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.*

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

- ***Accuracy Condition**: If $\hat{y}_t \neq \perp$, the output must 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 a function of $1/\epsilon$, $1/\delta$, 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 we call the hypothesis class $\mathcal{H}$ **efficiently KWIK-learnable**.*

Given below is a figure that shows the running of a KWIK algorithm as described by Li [2009].

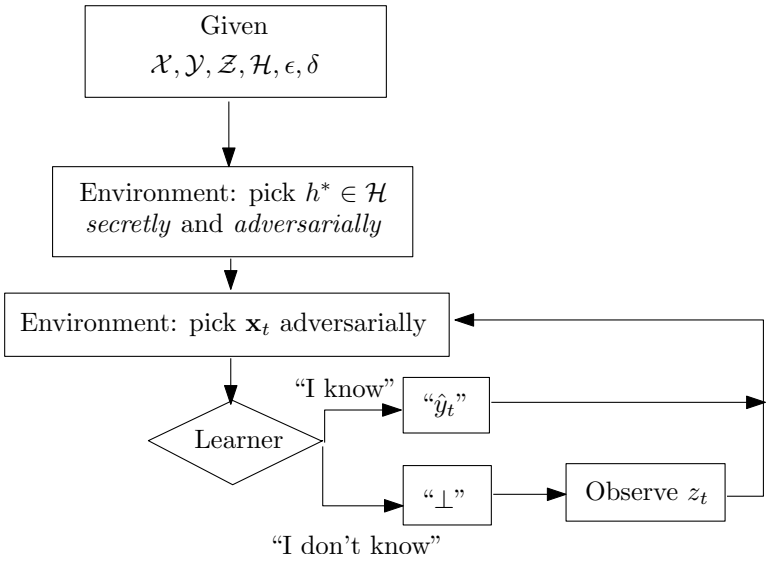


Figure 2.1: KWIK Protocol

The KWIK bound of an algorithm can be seen as a measure of how well the algorithm generalizes from experience in the input space under the constraint that the generalization must be accurate. The running of a KWIK algorithm can be visualized as shown in Figure 2.2.

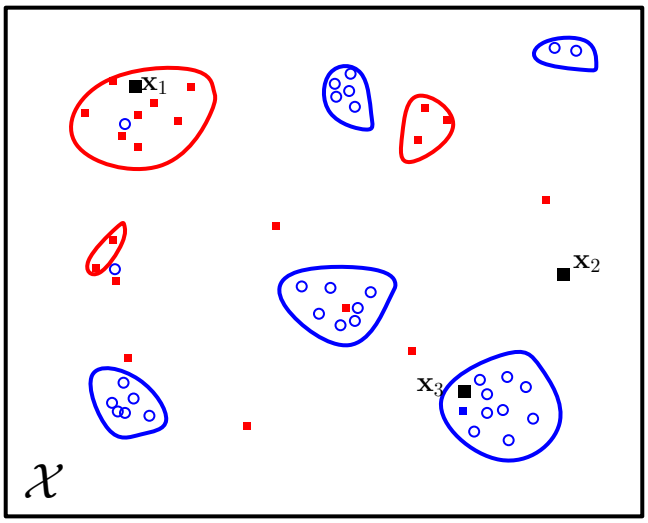


Figure 2.2: Generalization in KWIK

The algorithm successfully generalizes in the areas that are marked by red and blue boundaries. When an input point does not fall within these boundaries, the algorithm does not predict a label, but instead asks the expert for the true label, and accordingly changes these boundaries. That is, when the new input point is $\mathbf{x}_1$, the learner predicts *red*, on $\mathbf{x}_3$ it predicts *blue*, but on $\mathbf{x}_2$ it produces a *don't know*.

CHAPTER 3

RELATED WORK

In the following section we provide a complete survey of all the works that are related to the problem we are dealing with. We begin by discussing theoretical frameworks that are related to KWIK in Section 3.1. We also discuss two KWIK algorithms for online linear regression and compare them. Insights from this will be useful later when we discuss KWIK classification as part of our solution. Next, in Section 3.2 we discuss various approaches to inverse reinforcement learning in the literature. Finally, we summarize an array of works that have in some ways looked at active imitation learning, either model-free or model-based.

3.1 Related Learning Frameworks

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

3.1.1 Probably Approximately Correct Learning

As described in the introductory chapter (Chapter 1), 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 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 D. By $\epsilon-$accurate we mean that the probability of the prediction going wrong on an input point drawn from D is less than ϵ. That is,

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

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 hypotheses have 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 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 (MB) model, like KWIK, is appropriate for online learning. The difference here is that the learner is not required to be *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.

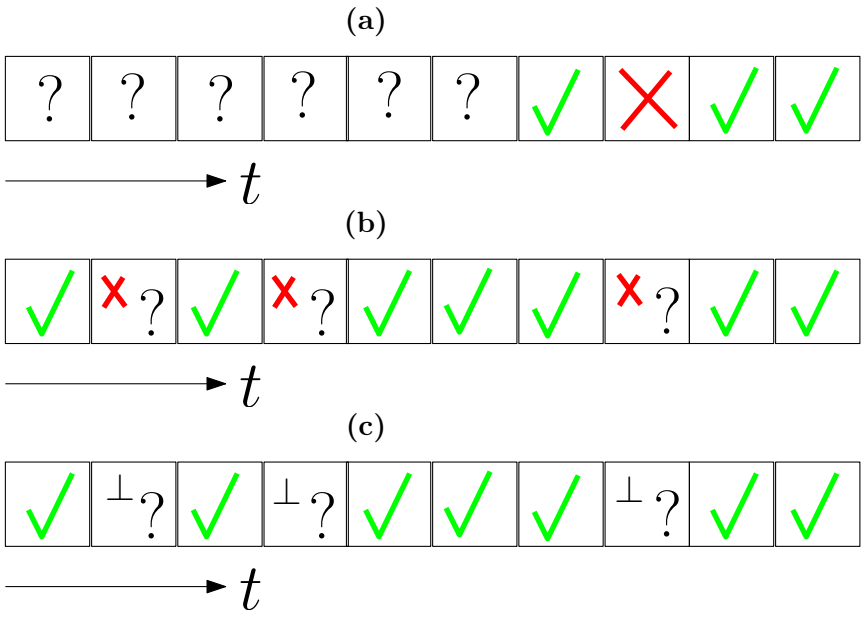


Figure 3.1: Visualization of the learning models a) PAC b) MB c) KWIK

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 know*s. 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 depends on this value. We will later see how our proposed KWIK-classification algorithm compares with the assumptions of this algorithm.

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. *However, the reader must be aware that it is not necessary to understand this section and the section that follows in complete detail to understand our contribution.*

In the linear regression problem, we assume that the input space is $\mathbb{R}^k$ and the output space is $\mathbb{R}$. For any input point $\mathbf{x} \in \mathbb{R}^k$, if an observation $z \in \mathbb{R}$ is provided by the teacher, 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.

3.1.5 Sketch of Theoretical Analysis

The details of the algorithm can be found in the Appendix, in Section 8.1. It is however not necessary to be aware of the technicalities of the algorithm to understand the following discussion. The KWIK bound of the algorithm (refer Algorithm 7) turns out to be

$\tilde{\mathcal{O}}\left(k^3/\epsilon^4\right)$ with appropriate parameter settings. It will be useful to understand how the KWIK online linear regression algorithm is analyzed to provide these guarantees. Li *et al.* [2011] present a detailed proof for the correctness of this algorithm under appropriate parameter settings.

We need to show that if the *preconditions* for the algorithm to make a valid prediction (refer Algorithm 7) hold good, then it can ***not*** be the case that the prediction that the algorithm makes is far off from the true answer.

The proof makes use of a lemma that if the preconditions are satisfied and yet the valid prediction was an inaccurate one, then *with high probability* , it will turn out that our estimate $\hat{\boldsymbol{\theta}}$ results in a greater error on the empirical observations provided by the teacher on the training data, when compared with the error of the true parameter. However, this contradicts the fact that the algorithm produces the parameters that minimizes the least squared error on the observations.

3.1.6 Selective Sampling

In this section, we discuss the work of Cesa-Bianchi and Orabona [2009] on 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, the guarantees of selective sampling are not as strong as the KWIK protocol.

A major difference between selective sampling in Cesa-Bianchi and Orabona [2009] and in our way of defining classification is that the latter makes 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 $\text{SGN}(\boldsymbol{\theta}_E^T \mathbf{x}_t)$. This makes the algorithm very similar to the linear regression where the expectation of the observation is a linear function of the parameter.

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:

$$\mathcal{O}\left(\frac{k}{\epsilon^2}\left(\ln\frac{T}{\delta}\right)\ln\frac{\ln(T/\delta)}{\epsilon}\right) \tag{3.3}$$

This we believe is a marked departure from the strong guarantees of KWIK. The advantage that this algorithm however provides over the KWIK linear regression (Algorithm 7) 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 the literature on the active learning approaches to IRL, and also active learning approaches to 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 $R : \mathcal{S} \to \mathbb{R}$ (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 defined on the space $\mathcal{S} \times \mathcal{A}$, which we do not present here. Thus, the reader must be careful not to confuse the following discussion with that for the case where rewards are defined for each state-action pair.

Theorem 1 *If* $\mathbf{R} \in \mathbb{R}^{|\mathcal{S}|}$ *is 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* $\forall \pi' \neq \pi$*:*

$$(\mathbf{T}_\pi - \mathbf{T}_{\pi'})(\mathbf{I} - \gamma \mathbf{T}_\pi)^{-1}\mathbf{R} \succcurlyeq 0 \tag{3.4}$$

The above theorem can be proved by making use of the following expression for the value functions that can be derived from the Bellman Equations described in Section 2.2.1:

$$\mathbf{V}^\pi = (\mathbf{I} - \gamma \mathbf{T}_\pi)^{-1}\mathbf{R} \tag{3.5}$$

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

$$\mathbf{T}_\pi \mathbf{V}^\pi \succcurlyeq \mathbf{T}_{\pi'} \mathbf{V}^\pi \tag{3.6}$$

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, $\mathbf{R} = \mathbf{0}$ 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 the authors 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:

$$\begin{aligned} \text{maximize} \quad & \sum_{s \in \mathcal{S}} \min_{a \in \mathcal{A}, a \neq \pi(s)} (\mathbf{T}_{\pi}^{s} - \mathbf{T}_{a}^{s})(\mathbf{I} - \gamma \mathbf{T}_{\pi})^{-1} \mathbf{R} - \lambda ||\mathbf{R}||_1 \\ \text{s.t.} \quad & (\mathbf{T}_{\pi} - \mathbf{T}_{\pi'})(\mathbf{I} - \gamma \mathbf{T}_{\pi})^{-1} \mathbf{R} \succcurlyeq 0 \qquad \forall \pi' \neq \pi \\ & ||\mathbf{R}||_{\infty} \leq R_{max} \end{aligned} \tag{3.7}$$

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 $\mathbf{\Phi}_R$ 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

$\hat{\boldsymbol{\theta}}^T \boldsymbol{\Phi}_R(s,a), \forall (s,a) \in \mathcal{S} \times \mathcal{A}$. With an abuse of notation, we have used $\pi(\hat{\boldsymbol{\theta}})$ to denote the dependence of the policy on the parameter setting. Next, every policy $\pi(\hat{\boldsymbol{\theta}})$ corresponds to what is called a feature expectation vector:

$$\boldsymbol{\mu}_{\pi(\boldsymbol{\theta})} = \mathbb{E}_{s_0 \sim D}[\sum_{t=0}^{\infty} \gamma^t \boldsymbol{\Phi}_R(s_t, a_t) | \pi] \in \mathbb{R}^k \tag{3.8}$$

where D is the initial state distribution. Hence, every parameter configuration $\hat{\boldsymbol{\theta}}$ 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 *expected* value of a policy:

$$\boldsymbol{\theta}_E^T \boldsymbol{\mu}_{\pi(\hat{\boldsymbol{\theta}})} = \mathbb{E}_{s \sim D} \left[V_{\pi(\boldsymbol{\theta}_E)}(s) \right] \tag{3.9}$$

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:

$$\mathbb{E}_{s_0 \sim D} \left[V_{\pi(\boldsymbol{\theta}_E)}(s) \right] - \mathbb{E}_{s_0 \sim D} \left[V_{\pi(\hat{\boldsymbol{\theta}})}(s) \right] \leq \epsilon \tag{3.10}$$

Here, $\mathbb{E}_{s_0 \sim D} \left[V_{\pi(\hat{\boldsymbol{\theta}})} \right] = \boldsymbol{\theta}_E \boldsymbol{\mu}_{\pi(\hat{\boldsymbol{\theta}})}$. 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 policy on the true parameter $\boldsymbol{\theta}_E$. This is done by ensuring that the corresponding feature expectation is $\epsilon-$approximate to the expert's feature expectation $\boldsymbol{\mu}_{\pi(\boldsymbol{\theta}_E)}$ which is assumed to be learned accurately from the demonstrations. That is:

$$||\boldsymbol{\mu}_{\pi(\hat{\boldsymbol{\theta}})} - \boldsymbol{\mu}_{\pi(\boldsymbol{\theta}_E)}||_2 \leq \epsilon \tag{3.11}$$

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. This is evident from the following:

$$\begin{aligned}
\mathbb{E}_{s_0 \sim D}\left[V_{\pi(\hat{\boldsymbol{\theta}})}(s)|\pi(\hat{\boldsymbol{\theta}})\right] - \mathbb{E}_{s_0 \sim D}\left[V_{\pi(\boldsymbol{\theta}_E)}(s)|\pi(\boldsymbol{\theta}_E)\right] &= |\boldsymbol{\theta}_E^T \mu_{\pi(\hat{\boldsymbol{\theta}})} - \boldsymbol{\theta}_E^T \mu_{\pi(\boldsymbol{\theta}_E)}| \\
&\leq ||\boldsymbol{\theta}_E||_2 ||\mu_{\pi(\hat{\boldsymbol{\theta}})} - \mu_{\pi(\boldsymbol{\theta}_E)}||_2 \\
&\leq 1 \cdot \epsilon = \epsilon
\end{aligned} \tag{3.12}$$

The algorithm (refer Algorithm **??** in Appendix) 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 sample complexity of this algorithm which is of interest to us is:

$$\frac{2k}{(\epsilon(1-\gamma))^2} \ln\left(\frac{2k}{\delta}\right) \tag{3.13}$$

Though the algorithm requires only $k \ln k$ samples where k is the dimensionality of the state representation, the solution does not consider an 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 any 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.

We provide a brief sketch of Bayesian IRL now. 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:

$$Pr(\mathcal{D}|\mathbf{R}) = \prod_{(s,a)\in\mathcal{D}} Pr((s,a)|\mathbf{R}) \tag{3.14}$$

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:

$$Pr(\mathbf{R}|\mathcal{D}) = \frac{Pr(\mathcal{D}|\mathbf{R})Pr(\mathbf{R})}{Pr(\mathcal{D})} \tag{3.15}$$

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 produce a distribution over the different paths which maximizes the likelihood of the demonstrations as parametrized above. 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 can be approached in two different ways: the *model-free* 'supervised learning' approach and the *model-based* inverse reinforcement learning approach. We will examine heuristics that have been proposed to solve either of these problems. However, theoretical analysis has 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. The reader is strongly encouraged to appreciate the technical content in this discussion.

3.3.1 Active Model-free Imitation Learning

We saw that in model-free imitation learning, the learning agent attempts to label states with actions using *supervision* from the expert's demonstrations. A multi-label classifier examines inputs 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 as discussed in Section 3.1.1.

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} [1]. A PAC learner would provide a learned policy $\hat{\pi}$ guaranteeing that with high probability of at least $1-\delta$, when a state is picked according to d_{π_E}, the probability of executing the wrong action is less than ϵ. But there is a catch here: the learning agent will be following the policy $\hat{\pi}$ which it learned, and not π_E itself. Thus, the error the agent commits while executing what it learned, is going to be the probability of executing a wrong action given that the state is picked from $d_{\hat{\pi}}$ *and not* d_{π_E}. 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 $\hat{\pi}$ - results in mistakes of the order of $T\epsilon^2$, where T is the episode length.

Ross and Bagnell [2010] propose the *forward training algorithm* (Algorithm 1) to address this issue. In the forward training algorithm, which is a passive imitation learning algorithm, a passive learner learns the policy for the t^{th} timestep iteratively from the distribution over states that is generated by the policy learned for the $(t-1)^{th}$ timestep. The authors show that the worst case guarantees for this algorithm is still the same as the traditional approach i.e., $T^2\epsilon$. However there are conditions where practically this

[1] Earlier we had used $\pi(\boldsymbol{\theta}_E)$ to denote the expert's policy. For the sake of presentation, we use π_E to denote the expert's policy and $\hat{\pi}$ the learner's estimate of the same.

algorithm outdoes the naive approach.

Algorithm 1 Forward Training Algorithm

Require: MDP M

Initialize $\pi_1^0, \ldots \pi_T^0$

for t in $0 \ldots T$ **do**

Sample many T-step trajectories using $\pi_1^t, \pi_2^t \ldots \pi_T^t$

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 π_i^i

$\forall j \neq i$, $\pi_j^i = \pi_j^{i-1}$

end forreturn $\pi_1^T, \ldots \pi_T^T$

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 as 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 more cleverly with this knowledge.

The authors show that using an active learner in place of a passive learner in the conventional approach of Syed and Schapire [2010] does not help because to learn the policy for timestep t, we must make t queries to the expert to generate the state distribution of the policy at the end of timestep $t-1$ which will be costly. Instead, we will follow the forward training approach to use the policy that the agent learns for timestep t to generate distribution for learning the policy for the next timestep. 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

$$\hat{\pi}_t = L_a(\epsilon, \frac{\delta}{T}, d^t_{\hat{\pi}_{t-1}}) \tag{3.16}$$

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*. For 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 consider a bayesian approach to solving the problem. Thus, the algorithm maintains a posterior distribution over the possible policies. If this posterior distribution tells us that a state is very less likely to be reached, the learner assumes that the teacher will report 'bad' with high probability. Knowing this, the learner avoids querying on such states.

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]. The authors 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 - which is similar to the *mistake bound* model.

Also very similar to this work on interactive policy learning is the dogged learning framework for robots of Grollman and Jenkins [2007] which also provides a protocol for teacher intervention and selective and autonomous execution of the policy. 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 2), 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 τ_{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.

Algorithm 2 Confidence-Based Autonomy algorithm

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 τ_{dist}, τ_{conf}
        Execute a
    end if
end while

The corrective demonstration component of the algorithm (which is not presented in Algorithm 2) 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 cor-

rected. The 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.

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. We believe that this cost parametrization is essentially $\mathbf{\Phi}_Q$.

To query for knowledge that is novel and *unfamiliar*, 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 or the *ambiguity* in the knowledge, we 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 states and update its inference about the rewards.

Algorithm 3 Active Inverse Reinforcement Learning Protocol

Require: Prior demonstration $\mathcal{D}$

while True **do**

Estimate posterior $Pr(\mathbf{R}|\mathcal{D})$

Compute $\hat{H}(s)$, the entropy over the policy at s, $\forall s \in \mathcal{S}$

Query expert for action at $s^* = \arg\max_s \hat{H}(s)$

Expert returns a^*

$\mathcal{D} = \mathcal{D} \cup \{(s^*, a^*)\}$

end while

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 a set of demonstrations. Lopes *et al.* [2009] use this effectively to evaluate the need for querying on a state. We might be tempted to claim 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; two very different reward settings could still lead to the same policy. Thus, the state

with the most entropy over the action with respect to the posterior reward distribution is the state that has to be queried on.

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 where KWIK provides us strong guarantees.

Secondly, even if the high entropy at a state is consistent with the 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 one which is so dissimilar from all the other states that we are not able to generalize further. However, we must note that this is not something that can be guaranteed by the KWIK framework as we will be forced to query on such a state if we have encountered it.

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 is an abstraction of the process of generalization and in turn provides $(\epsilon, \delta)-$guarantees.

The generalization that happens in the model-free approach to imitation learning is poorer when compared to that inherent to IRL because IRL algorithms make use of the structure of the MDP. The parametrization of rewards encapsulates the process of generalization across unseen states neatly. 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 with these trajectories as input.

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* that is presented by Sorg and Singh [2009]. Homomorphism between two MDPs is a way

of modeling equivalences between the states of the MDPs. This formulation helps us transfer a policy on one MDP to an equivalent policy in another MDP. However, conventionally homomorphism between two MDPs is computed using 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. Their idea 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

Before we end this section, 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 models.

The work is however 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 would be of interest to study active learning approaches to imitation learning.

In active learning, the learner actively chooses points from a pool of unlabeled data, and requests the expert for the label. In model-free active imitation learning techniques, the states form 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. They also analyze the performance of these algorithms with respect to the 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] (Section 3.3.1.2) address this by allowing the learner to *interactively* 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 theoretically be the best parameter to choose for the confidence thresholds that used by this algorithm as the approach that the authors propose is empirical.

In the light of the above drawbacks in earlier work on this problem, we present our contributions.

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*.

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.

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 the *selective sampling* algorithm studied by Cesa-Bianchi and Orabona [2009], Dekel *et al.* [2012] (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* $\mathcal{H} = \{\boldsymbol{\theta} | \boldsymbol{\theta} \in \mathbb{R}^k, ||\boldsymbol{\theta}||_2 = 1\}$ *that pass through the origin. The input space is defined to be* $\mathcal{X} = \{\mathbf{x} | \mathbf{x} \in \mathbb{R}^k, ||\mathbf{x}||_2 = 1\}$. *If the adversary picks a* $\boldsymbol{\theta}_E \in \mathcal{H}$, *the correct label of* $\mathbf{x} \in \mathbb{R}^k$ *is given by* $SGN[\boldsymbol{\theta}_E^T x] \in \{+1, -1\}$. *For the learner to be admissible, the following must hold good for every run, with probability at least* $1 - \delta$ *:*

- *Whenever the algorithm makes a prediction on* $\mathbf{x}$, *if* $|\boldsymbol{\theta}_E^T \mathbf{x}| > \epsilon$, *then* $\hat{h}(\mathbf{x}) = SGN[\boldsymbol{\theta}_E^T \mathbf{x}]$
- *The number of time-steps for which the algorithm emits* $\perp$ *is bounded by* $B(\epsilon, \delta, k)$ *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 at 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 $|\boldsymbol{\theta}_E^T \mathbf{x}| = ||\boldsymbol{\theta}_E||_2 ||\mathbf{x}||_2 \cos\phi = 1 \cdot 1 \cdot \cos\phi$ corresponds to the cosine of the angle ϕ between the normal of the hyperplane and the point $\mathbf{x}$ itself. When this value is low, it means that the point lies nearly orthogonal to the normal is hence very close to the separating hyperplane. 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 equation in the accuracy condition to the following:

$$|\boldsymbol{\theta}_E^T \mathbf{x}| > \epsilon ||\mathbf{x}||_2 \tag{4.1}$$

This may be compared to the KWIK-MB model proposed by Sayedi *et al.* [2010] who also assume that the inputs lie on the unit hypersphere and that the separating hyperplane passes through the origin. However, they assume that the examples from both classes are separated by a width of γ. Furthermore, their KWIK algorithm is allowed to make a fixed number of mistakes anywhere in the space. 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.

Assumption 2 *For the KWIK binary classification described above, we assume a teacher to be ϵ_Z-optimal, if the teacher outputs z for an input $\mathbf{x}$ such that:*

$$\begin{aligned} \mathbb{E}[z] &> \epsilon_Z \quad \text{if } \boldsymbol{\theta}_E^T \mathbf{x} \geq 0 \\ \mathbb{E}[z] &< -\epsilon_Z \quad \text{if } \boldsymbol{\theta}_E^T \mathbf{x} < 0 \end{aligned}$$

In other words, we expect that for any input point, the expert labels it correctly with

probability at least $1/2 + \epsilon_Z/2$. A good teacher will of course have a high ϵ_Z. This is a possibly relaxed assumption when compared to the selective sampling approach of Cesa-Bianchi and Orabona [2009] where the authors 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 algorithms 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_Z is the noise parameter of the teacher.

Throughout the discussion we will follow this convention: the points belonging to negative class will be represented by red squares and the points belonging to the positive class will be represented by blue rings.

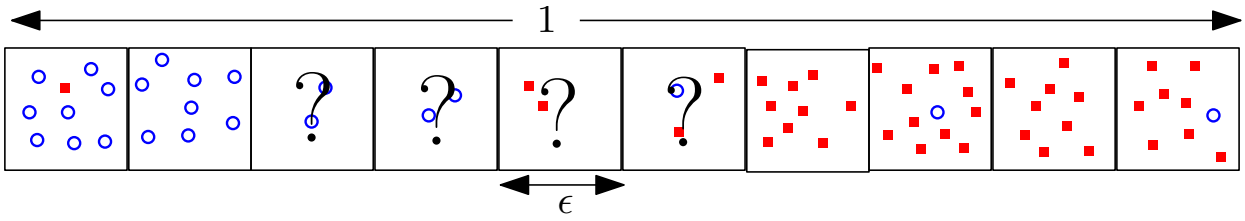


Figure 4.1: Visualization of the naive 1D classification

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 $\mathcal{O}\left(\frac{8}{\epsilon_Z^2}\ln(\frac{1}{\epsilon\delta})\right)$ (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 $1/2 + \epsilon_Z/2 - \epsilon_Z/4 > 1/2$ with a high probability of $1 - \epsilon\delta$ (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 $\mathcal{O}(\frac{1}{\epsilon\epsilon_Z^2}\ln(\frac{1}{\epsilon\delta}))$.

We have chosen segment sizes of length $1/\epsilon$ because each segment is finally assigned a single label and if the segment containing the separating point was of a greater size we will be generalizing a single label beyond the mistake region which would fail our KWIK conditions.

Remark 2 *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 did not make this assumption just to illustrate the situation.*

Remark 3 *One might suspect that the above solution is inefficient in that we do not generalize beyond the small space of width ϵ. That is, once we have two cells identified as belonging to the positive and negative class respectively, we could incorporate the idea that this knowledge can be generalized to all squares belonging to the extremes of this space (beyond these two squares). While this might practically result in a better algorithm, its performance in an adversarial setting is still the same. The teacher can still position the points in a manner that this generalization never occurs.*

4.4 KWIK Inverse Reinforcement Learning Protocol

We now present the reduction of IRL to KWIK classification.

Assumption 3 *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 4 $\forall (s,a) \in \mathcal{S} \times \mathcal{A},\ ||\mathbf{\Phi}_Q(s,a)||_2 \leq 1$

The above assumption is valid because any parametrization that we begin with can be made to satisfy this by scaling all the parameters by the same factor *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 $\text{SGN}[\boldsymbol{\theta}_E^T(\mathbf{\Phi}_Q(s,a) - \mathbf{\Phi}_Q(s,a'))]$ given $(\mathbf{\Phi}_Q(s,a) - \mathbf{\Phi}_Q(s,a'))$ as input.

Assumption 5 *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 $\mathbf{\Phi}_Q(s,a) - \mathbf{\Phi}_Q(s,a')$ to

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

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

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

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

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

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

We will now formalize the notion that if one action is much better than the other, and the classifier predicts an output, it will predict an output that favors the better action.

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

then:

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

$$\boldsymbol{\theta}_E^T(\mathbf{\Phi}_Q(s,a) - \mathbf{\Phi}_Q(s,a')) \geq 2\epsilon_C \tag{4.3}$$

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

and if

$$\boldsymbol{\theta}_E^T(\boldsymbol{\Phi}_Q(s,a') - \boldsymbol{\Phi}_Q(s,a)) \geq 2\epsilon_C \tag{4.4}$$

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

Proof If

$$\boldsymbol{\theta}_E^T(\boldsymbol{\Phi}_Q(s,a) - \boldsymbol{\Phi}_Q(s,a')) \geq 2\epsilon_C \tag{4.5}$$

then

$$\begin{aligned} \boldsymbol{\theta}_E^T \frac{(\boldsymbol{\Phi}_Q(s,a) - \boldsymbol{\Phi}_Q(s,a'))}{||\boldsymbol{\Phi}_Q(s,a) - \boldsymbol{\Phi}_Q(s,a')||_2} &> 2\epsilon_C \times \frac{1}{||\boldsymbol{\Phi}_Q(s,a) - \boldsymbol{\Phi}_Q(s,a')||_2} \\ &\geq 2\epsilon_C \times \tfrac{1}{2} = \epsilon_C \end{aligned}$$

The first inequality here comes from the assumption we made immediately before that. The second inequality comes from Lemma 1. Thus, the dot product between the normalized input and the true weight vector is strictly greater than ϵ_C 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. ■

We now formalize the notion that if the classifier favored one action over the other, the favored action cannot be much worse than the other action.

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

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

$$\boldsymbol{\theta}_E^T(\boldsymbol{\Phi}_Q(s,a) - \boldsymbol{\Phi}_Q(s,a')) \geq -2\epsilon_C \tag{4.6}$$

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

$$\boldsymbol{\theta}_E^T(\boldsymbol{\Phi}_Q(s,a') - \boldsymbol{\Phi}_Q(s,a)) \geq -2\epsilon_C \tag{4.7}$$

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

$$\boldsymbol{\theta}_E^T(\boldsymbol{\Phi}_Q(s,a) - \boldsymbol{\Phi}_Q(s,a')) < -2\epsilon_C$$

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 $2\epsilon_C$. 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 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.

4.4.1 Linear Search Based KWIK IRL Protocol

We present below a protocol followed by the learner in order to choose the $\epsilon-$optimal action if it takes a decision autonomously. The intuition is that the learner scans through every action once and compares it against the current candidate for the best action and updates the candidate.

Algorithm 4 Linear Search Based KWIK Inverse Reinforcement Learning Protocol

Require: Teacher $\mathcal{T}(\epsilon_Z)$ 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 $\boldsymbol{\Phi}_Q(s, a_i) - \boldsymbol{\Phi}_Q(s, \hat{a}_{best})$ to $\mathcal{C}$
    if Output of $\mathcal{C} = \perp$ **then**
      Present $\boldsymbol{\Phi}_Q(s, a_i) - \boldsymbol{\Phi}_Q(s, \hat{a}_{best})$ to $\mathcal{T}$
      $\mathcal{T}$ outputs observation of SGN$[\boldsymbol{\theta}_E \cdot (\boldsymbol{\Phi}_Q(s, a_i) - \boldsymbol{\Phi}_Q(s, \hat{a}_{best}))]$ (Refer Assumption 2)
      $\mathcal{C}$ 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$[\hat{\boldsymbol{\theta}}_E \cdot (\boldsymbol{\Phi}_Q(s, a_i) - \boldsymbol{\Phi}_Q(s, \hat{a}_{best}))]$
      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 4, 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.

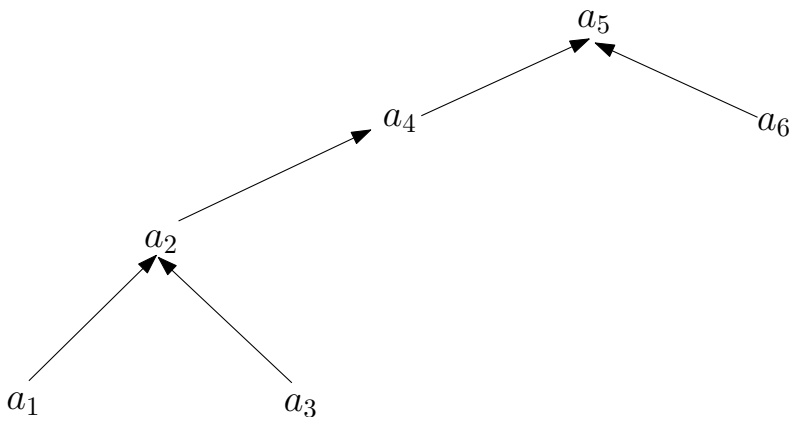


Figure 4.2: Example run of Linear Search

Example 1 *In figure 4.2, the classifier $\mathcal{C}$ favors a_2 over a_1; then the agent compares a_2 and a_3 and a_2 still remains the best seen so far. Next it scans a_4 and compares it with a_2. The algorithm proceeds similarly to pick a_5 eventually. Any $\perp$ produced by $\mathcal{C}$ is answered by the expert.*

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 induction hypothesis. That is, if $\hat{a}_i$ is the

candidate action picked by the algorithm, and a_i^* is the true 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_i^* must *still* be the best action amongst the first $i+1$ actions. However, from inequalities (4.8) and (4.9), we can see that:

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

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,

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

However, this implies a much weaker inequality:

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

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.*

4.4.2 Recursive Search Based KWIK IRL Protocol

In the above protocol we saw that the classifier is set to work with an accuracy parameter of $\frac{\epsilon}{2(|\mathcal{A}|-1)}$. We will now present an algorithm that works with an accuracy parameter of only $\frac{\epsilon}{2\log|\mathcal{A}|}$. The idea is that, instead of performing a linear scan over all the actions, the algorithm makes $\log|\mathcal{A}|$ iterations of pairwise queries and in each iteration it reduces the size of the candidate set of actions by half.

Assumption 6 $\exists H \in \mathbb{N}$ *such that* $|\mathcal{A}| = 2^{H-1}$

We make the above assumption only for the sake of simplicity.

Algorithm 5 Recursive Search Based KWIK Inverse Reinforcement Learning Protocol

Require: Teacher $\mathcal{T}(\epsilon_Z)$ with true weight vector for rewards $\boldsymbol{\theta}_E$, Admissible KWIK Classifier $\mathcal{C}(\frac{\epsilon}{2\log|\mathcal{A}|}, \delta)$ with weight estimate for rewards $\hat{\boldsymbol{\theta}}$

for $t = 1, 2, \ldots$ **do**
  s = Current State of the Environment
  Create a binary tree of $H = \log|\mathcal{A}| + 1$ levels with empty nodes
  Fill the nodes in the lowest level, level 1, with the actions in $\mathcal{A}$
  for i $= 2, \ldots H$ **do**
    for Node n in level i **do**
      $a_l \leftarrow$ action in left child of n
      $a_r \leftarrow$ action in right child of n
      $a \leftarrow a_l$
      Present $\boldsymbol{\Phi}_Q(s, a_r) - \boldsymbol{\Phi}_Q(s, a_l)$ to $\mathcal{C}$
      if Output of $\mathcal{C} = \perp$ **then**
        Present $\boldsymbol{\Phi}_Q(s, a_r) - \boldsymbol{\Phi}_Q(s, a_l)$ to $\mathcal{T}$
        $\mathcal{T}$ outputs observation of SGN$[\boldsymbol{\theta}_E \cdot (\boldsymbol{\Phi}_Q(s, a_r) - \boldsymbol{\Phi}_Q(s, a_l))]$ (Refer Assumption 2)
        $\mathcal{C}$ learns from output of $\mathcal{T}$ and updates $\hat{\boldsymbol{\theta}}$
        if Output of $\mathcal{T} = +1$ **then**
          $a \leftarrow a_r$
        end if

else

  $\mathcal{C}$ outputs SGN$[\hat{\boldsymbol{\theta}}_E \cdot (\boldsymbol{\Phi}_Q(s, a_r) - \boldsymbol{\Phi}_Q(s, a_l))]$

  if Output of $\mathcal{C} = +1$ **then**

    $a \leftarrow a_r$

  end if

end if

Assign a to Node n

end for

end for

Perform action at root node

end for

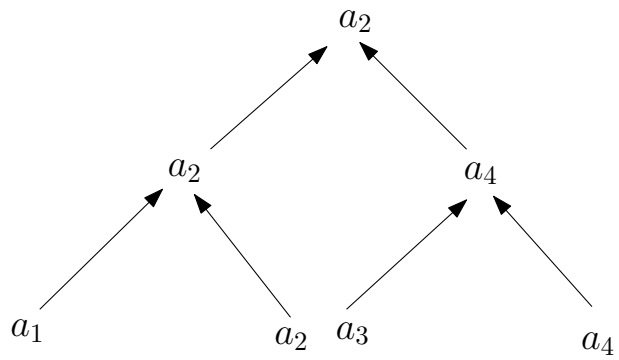


Figure 4.3: Example run of Recursive Search

Example 2 *In figure 4.3, the agent scans for the best action amongst the first set of actions a_1 and a_2, and the best amongst the next half a_3 and a_4. The classifier $\mathcal{C}$ favors a_2 and a_4 respectively. In the next round, the agent compares these two actions and learns that a_2 is the best it can choose. Any $\perp$ produced by $\mathcal{C}$ is answered by the expert.*

Theorem 5 *The action at a node on level i in the above algorithm is $(i-1)\dfrac{\epsilon}{\log|\mathcal{A}|}-$optimal with respect to the set of actions present in the leaves of the subtree rooted at the node, if no queries were made in the subtree.*

Proof The above statement is trivially true for level $i = 1$. We will use the induction hypothesis that the statement is true for all $i < H$. Now for a node at level $(i + 1)$ assume that the action chosen by its right child subtree is a_r and the left child subtree is a_l. Without loss of generality assume that the best action in the subtree rooted at this node is a_i^* and it belongs to the left subtree.

Since no query was made while choosing between a_l and a_r, we know that the chosen action was favored by the classifier $\mathcal{C}$. If the chosen action was a_l, we know from the induction hypothesis that it is $(i-1)\frac{\epsilon}{\log|\mathcal{A}|}-$optimal with respect to all the actions in the left subtree. However, a_i^* - the best action in the whole subtree - belongs to this left subtree. Hence, a_l satisfies the weaker condition that it is $i\frac{\epsilon}{\log|\mathcal{A}|}-$optimal with respect to a_i^* (and hence with respect to all the actions in the whole subtree).

Assume it is the case that $\mathcal{C}$ favored a_r over a_l. Hence, from Theorem 3 we know that

$$\boldsymbol{\theta}_E \cdot \boldsymbol{\Phi}_Q(s, a_r) \geq \boldsymbol{\theta}_E \cdot \boldsymbol{\Phi}_Q(s, a_l) - \frac{\epsilon}{\log|\mathcal{A}|} \tag{4.11}$$

From the induction hypothesis for the left child subtree we have that:

$$\boldsymbol{\theta}_E \cdot \boldsymbol{\Phi}_Q(s, a_l) \geq \boldsymbol{\theta}_E \cdot \boldsymbol{\Phi}_Q(s, a_i^*) - (i-1)\frac{\epsilon}{\log|\mathcal{A}|} \tag{4.12}$$

Combining the above two equations we will get:

$$\boldsymbol{\theta}_E \cdot \boldsymbol{\Phi}_Q(s, a_r) \geq \boldsymbol{\theta}_E \cdot \boldsymbol{\Phi}_Q(s, a_i^*) - i\frac{\epsilon}{\log|\mathcal{A}|} \tag{4.13}$$

Hence, in this case too we have that the action favored by the classifier is $i\frac{\epsilon}{\log|\mathcal{A}|}-$optimal with respect to a_i^* (and hence with respect to all the actions in this subtree).

■

Corollary 2 *If the algorithm did not make any queries in choosing an action at a state, then the chosen action is $\epsilon-$optimal.*

The above claim follows from the fact that the level of the root is $H = \log|\mathcal{A}| + 1$ and the action at the root is $(H-1)\frac{\epsilon}{\log|\mathcal{A}|}-$optimal.

4.5 Learner-Expert Query Interface

In the discussion of the linear and recursive search based protocols (Algorithms 4 and 5), we have assumed that the learner can pose pairwise queries to the expert and acquire knowledge about any individual state-action pair. In practice, this might however be infeasible. Often, the teacher might be in a position to rather tell us what the best action is; the teacher might not have knowledge about the complete ordering of the actions (noisy or otherwise).

We will have to overcome this by designing an algorithm that can still extract information from such a teacher while being in a position to provide guarantees on the KWIK bound. The linear search protocol is not in a position to handle this. For example, when the algorithm tries to compare action a_i with $\hat{a}_{i-1}$ (the candidate from the first $i-1$ actions) and the classifier produces a *don't know*, and it requests the teacher to tell it what the best action is at the state, the teacher might simply point out an action that is neither a_i nor $\hat{a}_{i-1}$ to be the best. Hence, we do not know which of a_i and $\hat{a}_{i-1}$ is the better action i.e., the *don't know* of the learner is not taken care of! Thus, the KWIK guarantees of the algorithm is no longer valid.

We will now discuss a modification of the linear search based protocol that can adopt to this situation. When the algorithm encounters a *don't know* while comparing $\hat{a}_{best}$ with some other action, it keeps searching the remaining candidate space of actions $\mathcal{A}'$ for a pair with whom comparing the current best estimate does not produce a *don't know*. If it successfully finds a pair, the algorithm chooses the better of the two, eliminates the other, and proceeds as earlier. However, if it does not find a pair, it queries the teacher for the best action local to $\mathcal{A}' \cup \{\hat{a}_{best}\}$. But note that every action in this set produced a *don't know* when compared with at least one other action in the set. Hence, when the teacher answers this query, we can be certain that at least one of the *don't knows* is answered.

To understand how the KWIK bound of the algorithm changes, we have to view the process as a posing a *batch* of *don't know*s to the expert of which at least one will be answered. The batch can be of size $\mathcal{O}(|\mathcal{A}|)$ as it corresponds to the $\perp$'s generated while comparing $\hat{a}_{best}$ with to at most $|\mathcal{A}|-1$ actions. Thus, the KWIK bound of the algorithm is expected to be multiplied by a factor of $\mathcal{O}(|\mathcal{A}|)$.

While the above approach results in posing exactly one query to the expert at a given state, we would still want to relax the requirement that the expert must be able to know which action is the best amongst a set of actions. We would ideally want the teacher to be aware of only the best action or perhaps the set of near-optimal actions at a state.

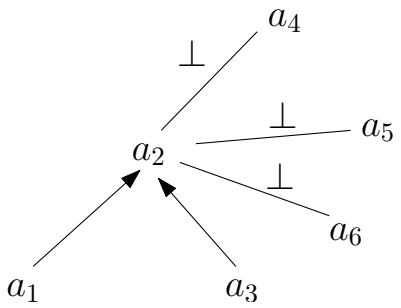


Figure 4.4: Example run for Modified Linear Search

Example 3 *In figure 4.4, the algorithm has scanned the first three actions and has cur-*

rently chosen a_2 to be the best. However, when it compares a_2 to the next action a_4 it encounters a $\perp$ from the classifier $\mathcal{C}$. While the simple linear search algorithm would have directed this query to the expert, this algorithm first searches the rest of the action space for an action with which $\mathcal{C}$ can compare a_2. Only if none of the actions are comparable (all produce a don't know), does it ask the expert, which of a_2, a_4, a_5 and a_6 is the best action. Observe that any answer that the expert replies will answer one of the don't know queries.

4.6 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 approach 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 smaller surfaces/segments, and make predictions in these segment 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 (Union Bound Lemma 2). From Lemma 5, each segment will thus require $\frac{8}{\epsilon_Z^2}\ln\left(\frac{N}{\delta}\right)$ points to ensure that the majority of labels is correct. Thus, we will make as many queries as:

$$N \times \frac{8}{\epsilon_Z^2}\ln\left(\frac{N}{\delta}\right) \tag{4.14}$$

However, we must make sure that while splitting the hypersphere surface, the width of the segments is limited by the parameter ϵ in the KWIK condition (Refer Appendix

Section 9.1 for a detailed discussion of this relation). We could come up with many different ways of splitting the unit hypersphere surface that obey the above constraint. We skip discussing these ideas here as they do not form the core of our discussion. However, we must note that such a splitting is expected to be of the order $\mathcal{O}(1/\epsilon^k)$. Thus, we will end up making queries of the order:

$$\frac{1}{\epsilon^k \epsilon_Z^2} \ln\left(\frac{1}{\epsilon^k \delta}\right) \tag{4.15}$$

which is ***exponential*** in k, the dimensionality of the space!

Our aim in this discussion is to gain insights about how we could possibly *characterize* regions in the input space according to how confident the learner is about the class that the space belongs to. We resort to continuing the discussion comfortably for the 2D case; it is not completely clear as to how it would generalize to multiple dimensions.

While earlier we resorted to splitting up the space into small regions, which is naive because it does not lead to generalization, we will now look at a more sophisticated way of doing this. We must however note that the solution is still exponential if somehow generalized to multiple dimensions. We believe that with appropriate assumptions in the model this approach might *perhaps* lead us to a polynomial KWIK bound.

We will now define a few terms which will be required for the discussion that follows. As earlier, we will follow the convention that the points belonging to negative class will be represented by red squares and the points belonging to the positive class will be represented by blue rings. Also, we will present the figures in 2D for the sake of visualization. In Figure 4.5 AB is the estimated separating hyperplane $\hat{\boldsymbol{\theta}}$. While the input to the classifier is always normalized, we will present the inputs as unnormalized in the figures, again for the sake of visualization. The significance of the lines other than

AB will be discussed further.

We will use a failure probability of δ' in our discussion to distinguish it from δ, the global failure probability i.e., the probability of failure of the KWIK algorithm. δ' would have to be set accordingly so that applying union bound across all applications of this failure probability during the run leads to a failure of at most δ.

Definition 15 *We define $\mathcal{N}_{\epsilon'}(\boldsymbol{\theta})$, the $\epsilon'-$neighbourhood of $\boldsymbol{\theta}$ such that:*

$$\mathcal{N}_{\epsilon'}(\boldsymbol{\theta}) = \left\{ \boldsymbol{\theta}' \ s.t. \ ||\boldsymbol{\theta}'||_2 = 1, |\boldsymbol{\theta}^T \boldsymbol{\theta}'| \leq \sqrt{1-\epsilon'^2} \right\} \tag{4.16}$$

Assumption 7 *The estimated classifier is arrived at by minimizing the number of mistakes on the training datapoints $\mathcal{D}$. That is*

$$\hat{\boldsymbol{\theta}} = \arg \min_{\boldsymbol{\theta}:||\boldsymbol{\theta}||_2=1} \sum_{(\mathbf{x},z)\in\mathcal{D}} \mathbb{I}\left(SGN[\boldsymbol{\theta}^T \mathbf{x}] \neq z\right) \tag{4.17}$$

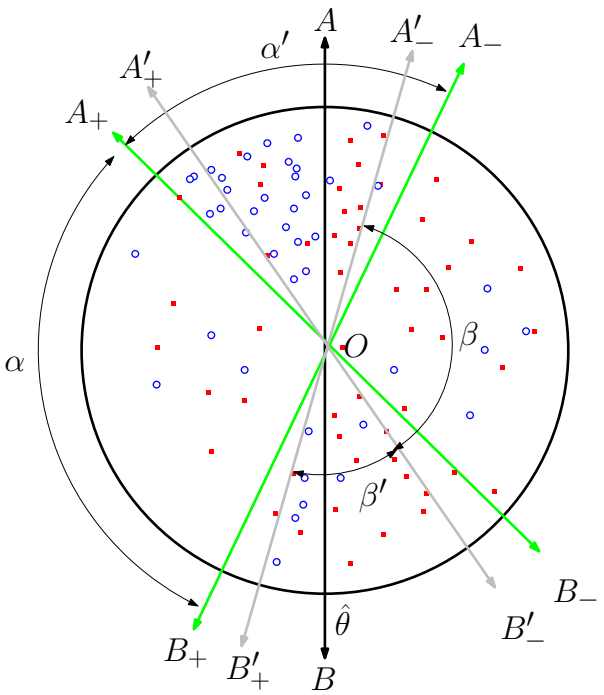


Figure 4.5: Visualization of the k-D classification

Assumption 8 *The region swept between A_+B_- and AB (along α'), and the region swept between A_-B_+ and AB (along α') respectively have at least $M = \frac{8}{\epsilon_Z^2} \ln \frac{1}{\delta'}$ queried points in them each.*

Definition 16 *The regions swept between A_+B_- and A_-B_+ (along α) are called the **known regions**. The rest of the space is considered **unknown**.*

We use the word *known* because we will see that the learner can identify the label of this space correctly with high probability.

Assumption 9 *The angle between $A'_-B'_+$ and A_-B_+, and the angle between $A'_+B'_-$ and A_+B_- are such that:*

$$|\mathbf{A'_-B'_+} \cdot \mathbf{A_-B_+}| = |\mathbf{A'_+B'_-} \cdot \mathbf{A_+B_-}| = \sqrt{1-\epsilon^2} \tag{4.18}$$

Thus, the pairs of lines belong to each other's $\epsilon-$neighborhoods.

Note that it is unclear as to how the above assumptions generalize to more dimensions.

Definition 17 *The regions swept between $A'_+B'_-$ and $A'_-B'_+$ (along β) are called the **safe regions**. The rest of the space is considered **unsafe**.*

We use the word *safe* because this space includes the *known* region which the learner can safely and correctly predict on; and also includes the 'mistake' region around the true separating hyperplane where the learner is allowed to predict safely but not necessarily correctly.

Based on the above assumptions, we make the following claim.

Theorem 6 *With high probability of at least* $1-\delta'$*, if the true separating hyperplane* $\boldsymbol{\theta}_E$ *lied in the* ***known region****, our estimate of the separating hyperplane would not be* AB*.*

Proof We begin by assuming that the true separating hyperplane is CD as shown in Figure 4.6. Here we assume a specific configuration of $+$ and $-$ as labeled by CD. However, this is without loss of generality, which will be evident from the manner of this proof.

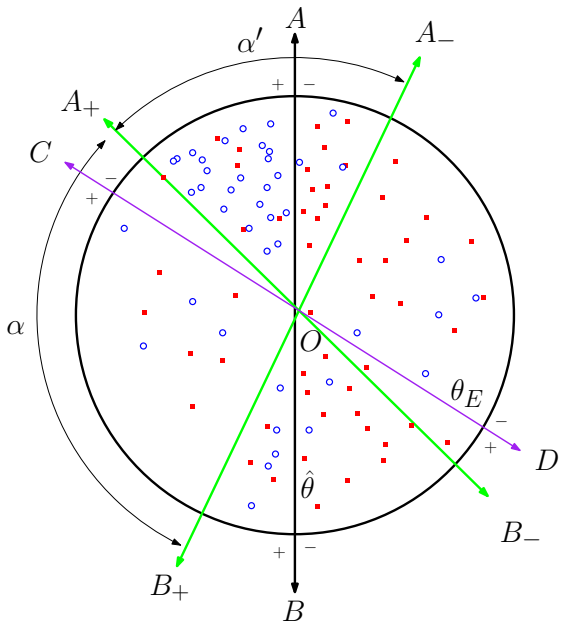


Figure 4.6: The true separating hyperplane assumed to be in the known region

We will show that if it were the case that CD was the true separating hyperplane, then empirically our estimated separating hyperplane will not be the result of the optimization in Assumption 7. In fact, A_+B_- will minimize the number of mistakes strictly better than our own estimate AB. In the following discussion, we will use the definition of M from Assumption 8.

Consider only the points in the region swept between AB and A_+B_- as shown in Figure 4.7. We will make use of the fact that if AB was perturbed anti-clockwise to A_+B_-, the only points that the new separating hyperplane labels differently are those

in this region. On the rest of the space the two separating hyperplanes concur on the labeling.

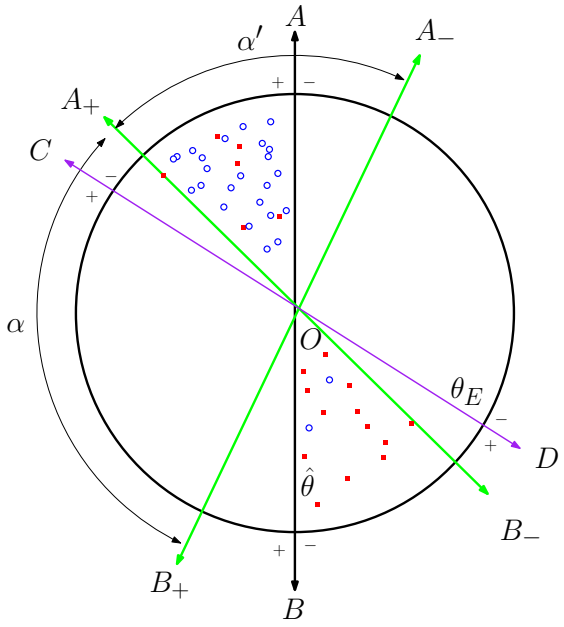


Figure 4.7: The points that the two separating hyperplanes disagree on

If CD were the true separating hyperplane, from Lemma 5 with high probability a majority of these M points will be 'correctly' labeled by the teacher where correctness equates to being consistent with the labeling of CD. More precisely, with high probability of at least δ', $M(1/2 + \epsilon_Z/4)$ points are labeled correctly by the teacher and at most $M(1/2 - \epsilon_Z/4)$ are labeled incorrectly.

From the way the separating hyperplanes AB, CD and A_+B_- are positioned we can observe the following: AB will now disagree with at least $M(1/2 + \epsilon_Z/4)$ labels in this region. On the other hand, A_+B_- will disagree with at most $M(1/2 - \epsilon_Z/4)$ labels in this region. However since the difference between these two separating hyperplanes in the number of disagreements on the training labels arises only from this space, we can claim that AB makes at least $M\epsilon_Z/2$ mistakes more than A_+B_- on the training data. This goes against Assumption 7 that AB was chosen to minimize the number of such

mistakes on the training labels. ■

Remark 4 *In the above proof we assumed that the true separating hyperplane had a particular configuration, and for that configuration a high probability theorem is applied. Thus if we want to consider each parameter setting of the true separating hyperplane in the known space, we might have to consider infinitely many such settings! While Strehl and Littman [2007] face a similar problem, they address it by discretizing the parameter space and considering the application of the high probability theorem only for those settings. Though this results in applying a failure of δ' exponentially many times, it affects the KWIK bound only polynomially because of the logarithmic factor on the failure parameter. However, we are not sure if this sort of an analysis would be applicable here as their discretization applies on the estimated separating hyperplane produced by the algorithm and not on the true separating hyperplane.*

While with little adjustments we might be able to address this problem in the same manner as that of Strehl and Littman [2007] we provide a different solution here. We claim that for any position of the true separating hyperplane in one half of the known region, the above proof applies 'at one go' i.e., instead of applying the Lemma 5 for each parameter setting, the proof seems applicable independent of the parameter setting as long as the true separating hyperplane lied in a specific half of the known region. We might however have to apply the proof again for the other half of the known region, for which the proof is identical. Hence, we may only incur a failure of $2\delta'$.

Corollary 3 *With probability at least $1 - 2\delta'$, the true separating hyperplane lies in the region swept between A_+B_- and A_-B_+ (along α').*

Corollary 4 *With high probability, the predictions made by the estimated separating hyperplane in the known region are correct.*

A crucial point here is that the algorithm need not make use of the estimated separating hyperplane's prediction only in the known region. There is a small region around this which the algorithm can still safely predict because in the worst case, this region will fall in the *mistake* region of the separating hyperplane i.e., the learner is allowed to make infinitely many mistakes here! Recall that we called this sum total of the known region and the mistake region as the safe region. The algorithm is summarized as follows.

Algorithm 6 KWIK Binary Classification Algorithm

Require: ϵ, δ, ϵ_Z

Compute δ' based on δ

Compute M from ϵ, δ' and ϵ_Z (Assumption 8)

for $t = 1, 2, \ldots$ **do**

Environment produces input $\mathbf{x}$

if $\mathbf{x}$ belongs to safe region **then**

Output SGN($\hat{\boldsymbol{\theta}}^T \mathbf{x}$)

else

Ask environment for label

Environment produces label z with noise as defined in Assumption 2

Learner updates $\hat{\boldsymbol{\theta}}$ using the objective in Assumption 7

Learner shrinks the unsafe region according to Definition 17

end if

end for

Remark 5 *How many times do we have to account for failure? We have to account*

for failure only when we update the safe/known regions. We do not have to account for failure on every prediction as they are not completely independent. All the predictions that happen in a specific configuration of the safe region are tied together by the same failure probability. We will thus account for only as many failures as there are queried points. Since the number of queried points is finite, the number of times we worry about failure is only finitely many.

Remark 6 *Practically, it might not be possible to solve the optimization problem in Assumption 7. We could however, possibly assume that we can always find a solution within the $\epsilon-$neighborhood of the true solution. The above proofs can be slightly adjusted to accommodate this but we do not deal with it here.*

Remark 7 *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.*

To understand the above remark, consider the space of input points which are not normalized. Visually, the space 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 centered at the origin, inscribed within this cone. 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, we *know* that the separating hyperplane has to pass through the origin, and any sort of perturbation will be only with respect to its 'angular' position; if the separating hyperplane is perturbed the space it sweeps is only a conical region and hence we must allow errors to occur only in this region.

We end this section by noting that the above algorithm, if at all extended clearly to multiple dimensions, will still be exponential in the number of dimensions. This is because, in the worst case, every time the safe region is expanded, it is expanded only by an amount that corresponds to a volume of an ϵ-neighborhood (Definition 15) of the current boundary between the safe and unsafe regions. However, we know that there are exponentially many such volumes that the safe region will have to cover.

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 considered 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; 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 groundwork for an algorithm that could work 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 a framework 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 would be to study algorithms for classification in the KWIK framework as we defined. We have provided the groundwork for the classification algorithm which we however understand to have exponential bounds. This could be a consequence of the extremely localized region in which mistakes are allowed. Perhaps with more relaxed requirements, that are 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. One issue in this aspect is the kind of queries that we assume can be posed to the teacher. We have considered pairwise queries and also queries about which action is the best of a subset of actions. However, 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 I

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*

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

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*

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

and

$$\mu \stackrel{\text{def}}{=} \mathbb{E}[\hat{\mu}]$$

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,

$$Pr\left(\hat{\mu} \leq 1/2 + \epsilon/4\right) \leq \exp\left(-m\epsilon^2/2\right) \tag{7.7}$$

Thus, when

$$m = \frac{8}{\epsilon^2} \ln\left(\frac{1}{\delta}\right) \tag{7.8}$$

we can ensure that

$$Pr(\hat{\mu} > 1/2 + \epsilon/4) > 1 - \delta \tag{7.9}$$

CHAPTER 8

APPENDIX II

8.1 KWIK Online Linear Regression

We provide here a summary of the details of the KWIK Linear Regression algorithm discussed in Strehl and Littman [2007].

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 $X_t \in \mathbb{R}^{m \times k}$ 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.

$$X^T X = U \Lambda U^T \tag{8.1}$$

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

$$\tilde{\mathbf{q}} \stackrel{\text{def}}{=} X\bar{U}\bar{\Lambda}^{-1}\bar{U}^T\mathbf{x}_t \tag{8.2}$$

$$\tilde{\mathbf{u}} \stackrel{\text{def}}{=} [0, \ldots, 0, \mathbf{u}_{n+1}^T\mathbf{x}_t, \ldots \mathbf{u}_k^T\mathbf{x}_t] \tag{8.3}$$

Note that $\bar{U} \in \mathbb{R}^{k\times n}$, $\bar{\Lambda} \in \mathbb{R}^{n\times n}$, $\tilde{\mathbf{q}} \in \mathbb{R}^{m\times 1}$ and $\tilde{\mathbf{u}} \in \mathbb{R}^{k\times 1}$.

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 7 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 (8.2), (8.3)

  if $||\tilde{\mathbf{q}}||_2 \leq \alpha_1$ and $||\tilde{\mathbf{u}}||_2 \leq \alpha_2$ **then**

    $\hat{\boldsymbol{\theta}} \leftarrow \arg\min_{\boldsymbol{\theta}\in\mathbb{R}^n : ||\boldsymbol{\theta}||_2 \leq 1} ||X\boldsymbol{\theta} - \mathbf{z}||_2^2$

    Learner outputs $\hat{\boldsymbol{\theta}}^T\mathbf{x}_t$

9: **else**

10: Learner outputs $\perp$

11: Environment produces observation z_t such that $\mathbb{E}[z_t] = \boldsymbol{\theta_E}^T \mathbf{x}_t$

12: Add $\mathbf{x}_t^T$ as a new row to X

13: Add z_t as a new element to vector $\mathbf{z}$

14: **end if**

15: **end for**

The proof makes use of a lemma that if the preconditions are satisfied and yet the valid prediction was an inaccurate one, our estimate is inaccurate with respect to the true outputs on the 'training data'. If our estimate is inaccurate on these true outputs, then *with high probability* $\hat{\boldsymbol{\theta}}$, it will turn out that our estimate results in a greater error on the empirical observations provided by the teacher on the training data, when compared with the error of the true parameter. However, this contradicts the fact that the algorithm uses the parameters that minimizes the least squared error on the observations.

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 $\Delta_E(\hat{\boldsymbol{\theta}}) \stackrel{\text{def}}{=} \sqrt{\sum_{i=1}^m \left((\boldsymbol{\theta}_E - \hat{\boldsymbol{\theta}})^T \mathbf{x}_i\right)^2}$, 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 ($\boldsymbol{\theta}_E^T \mathbf{x}_i$), and not with respect to the noisy observations of the training data (z_i).

The second step to the proof is showing that when this true error on the training data is sufficiently large, then the observed *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, $\sqrt{\sum_{i=1}^m (\boldsymbol{\theta}_E^T \mathbf{x}_i - z_i)^2}$ *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 $\boldsymbol{\theta}_E^T \mathbf{x}_i$.

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.

8.1.0.1 Proof for KWIK Bound

The first step towards bounding m, the number of *don't know*'s for this algorithm in terms of n, ϵ, δ 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 $\sum_t ||\tilde{\mathbf{q}}_t||_2$ and $\sum_t ||\tilde{\mathbf{u}}_t||_2$ in terms of m and n and by using the conditions of the algorithm which provide a lower bound on $||\tilde{\mathbf{q}}_t||_2$ and $||\tilde{\mathbf{u}}_t||_2$ 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 faced a similar situation in the KWIK classification algorithm.

8.2 Selective Sampling

The online classification algorithm provided by Cesa-Bianchi and Orabona [2009] maintains an estimate of the classifier as follows:

$$\hat{\boldsymbol{\theta}} \stackrel{\text{def}}{=} (I + S_{t-1}S_{t-1}^T + \mathbf{x}_t\mathbf{x}_t^T)^{-1} S_{t-1}\mathbf{Y}_{t-1} \tag{8.4}$$

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.

In Equation (8.4), 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.

$$\begin{aligned}
\mathbb{E}[\hat{\boldsymbol{\theta}}_t^T \mathbf{x}_t] &= \mathbb{E}[\hat{\boldsymbol{\theta}}_t^T]\mathbf{x}_t \\
&= \mathbb{E}[((I + S_{t-1}S_{t-1}^T + \mathbf{x}_t\mathbf{x}_t^T)^{-1} S_{t-1}\mathbf{Y}_{t-1})^T]\mathbf{x}_t \\
&= \mathbb{E}[\mathbf{Y}_{t-1}^T S_{t-1}^T \left((I + S_{t-1}S_{t-1}^T + \mathbf{x}_t\mathbf{x}_t^T)^{-1}\right)^T]\mathbf{x}_t \\
&= \mathbb{E}[\mathbf{Y}_{t-1}^T]S_{t-1}^T(I + S_{t-1}S_{t-1}^T + \mathbf{x}_t\mathbf{x}_t^T)^{-1}\mathbf{x}_t \\
&= \boldsymbol{\theta}_E^T S_{t-1}S_{t-1}^T(I + S_{t-1}S_{t-1}^T + \mathbf{x}_t\mathbf{x}_t^T)^{-1}\mathbf{x}_t \\
&= \boldsymbol{\theta}_E^T\mathbf{x}_t - \boldsymbol{\theta}_E^T(I + \mathbf{x}_t^T\mathbf{x}_t)(I + S_{t-1}S_{t-1}^T + \mathbf{x}_t\mathbf{x}_t^T)^{-1}\mathbf{x}_t \\
&= \boldsymbol{\theta}_E^T\mathbf{x}_t - B_t
\end{aligned} \tag{8.5}$$

where in the last line,

$$B_t = \boldsymbol{\theta}_E^T(I + \mathbf{x}_t\mathbf{x}_t^T)(I + S_{t-1}S_{t-1}^T + \mathbf{x}_t\mathbf{x}_t^T)^{-1}\mathbf{x_t} \tag{8.6}$$

Thus, B_t is an unknown value that denotes the *bias* in our estimate of the point's label.

For the sake of simplicity, we will define $A_t \stackrel{\text{def}}{=} (I + S_{t-1}S_{t-1}^T + \mathbf{x}_t\mathbf{x}_t^T)$. We will have to define a few more terms which will be used in the conditions that will be checked for querying. The rest of the analysis involves bounding the value of B_t and from there bounding the probability of our estimate lying far away from the true estimate.

Algorithm 8 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 $\left[\epsilon - \mathbf{x}_t^T A_t^{-1}\mathbf{x}_t - ||A_t^{-1}\mathbf{x}_t||_2\right]_+ \leq ||S_{t-1}^T A_t^{-1}\mathbf{x}_t||_2 \sqrt{2\ln\left(\frac{t(t+1)}{\delta}\right)}$ **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 (8.4)

  else

    Predict $\text{SGN}(\hat{\boldsymbol{\theta}}_t^T\mathbf{x}_t)$

  end if

end for

8.3 IRL with Parametrized Rewards

We present here the algorithm provided by Abbeel and Ng [2004] for the MDP that is assumed to have linearly parametrized algorithms.

Algorithm 9 Apprenticeship Learning via Inverse Reinforcement Learning

Require: Expert's feature expections $\boldsymbol{\mu}_{\pi(\boldsymbol{\theta}_E)}$

$\pi_0 \leftarrow$ Randomly initialized policy

Candidate policy pool $\Pi \leftarrow \{\pi_0\}$

for $i = 1, 2, \ldots$ **do**

    $t_i \leftarrow \max_{\boldsymbol{\theta}:||\boldsymbol{\theta}||_2 \leq 1} \min_{\pi \in \Pi} \boldsymbol{\theta}^T (\boldsymbol{\mu}_{\pi(\boldsymbol{\theta}_E)} - \boldsymbol{\mu}_\pi)$

    $\boldsymbol{\theta}_i \leftarrow \boldsymbol{\theta}$ that corresponds to the above maximum

    Break if $t_i \leq \epsilon$

    Compute π_i optimal on $\boldsymbol{\theta}_i$

    $\Pi \leftarrow \Pi \cup \{\pi_i\}$

end for

CHAPTER 9

APPENDIX III

9.1 Width of the Hypersphere Segments

In this section we derive a lower bound on the required width of any segment on the unit hypersphere that is required for the naive KWIK classification algorithm that is discussed in Section 4.6. To begin with, we will note that we should not generalize across a segment through which two possible separating hyperplanes pass and both of them are not in each other's $\epsilon-$neighborhood. This is because we want the segment containing the true hyperplane to not have a size greater than the 'mistake region' which is equivalent to the $\epsilon-$neighborhood volume.

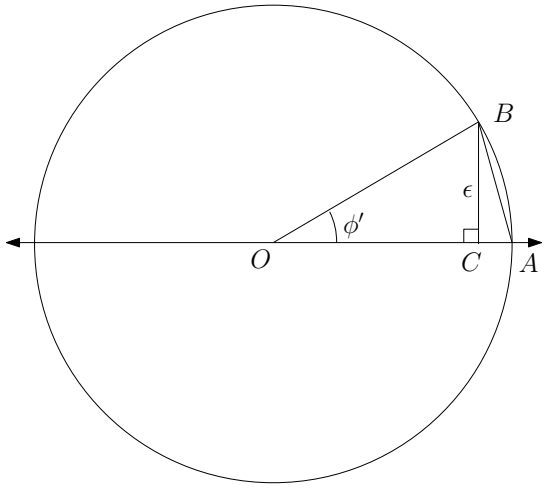


Figure 9.1: Constraint on the width of the segment

In the above figure we use ϕ' to distinguish it from the symbol ϕ that has been used

already to denote the angle between the input point $\mathbf{x}$ and the normal of the hyperplane $\boldsymbol{\theta}$. In fact, $\phi' = \pi/2 - \phi$. Thus if $cos\phi = \epsilon$, we would like $cos\phi' = \sqrt{1-\epsilon^2}$. The symbols A, B and C must not be confused with the notations in the earlier figures in Section 4.6.

The width of the hypersphere segment BA is the length of the line segment BA. We can see that:

$$2l_{OA}sin\left(\frac{\phi'}{2}\right) = l_{BA} \tag{9.1}$$

Since $cos\phi = 1 - 2\sin^2(\phi/2)$,

$$\sqrt{1-\epsilon^2} = 1 - 2\sin^2\left(\frac{\phi}{2}\right) \tag{9.2}$$

Combining the above two equations:

$$l_{BA} = \sqrt{2(1-\sqrt{1-\epsilon^2})} \tag{9.3}$$

Thus, the width of the segment must be $\Omega\left(\sqrt{2(1-\sqrt{1-\epsilon^2})}\right)$. To get a neater bound, we use the AM-GM inequality $2\sqrt{1(1-\epsilon^2)} \leq 1 + 1 - \epsilon^2$ to get

$$1 - \sqrt{1-\epsilon^2} \geq \frac{\epsilon^2}{2} \tag{9.4}$$

Thus, the width must be $\Omega(\epsilon)$.

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.

Dekel, O., **C. Gentile**, and **K. Sridharan** (2012). Selective sampling and active learning from single and multiple teachers. *J. Mach. Learn. Res.*, **13**(1), 2655–2697. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=2503308.2503327.

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. (2009). *A unifying framework for computational reinforcement learning theory*. Ph.D. thesis, Rutgers, The State University of New Jersey.

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.