BOSTON UNIVERSITY

COLLEGE OF ENGINEERING

Thesis

# SAFETY-AWARE APPRENTICESHIP LEARNING

by

**WEICHAO ZHOU**

B.S., Fudan University, 2012

Submitted in partial fulfillment of the

requirements for the degree of

Master of Science

2018

## Acknowledgments

I want to express my deepest appreciation to my advisor, Professor Wenchao Li. During my two-year master's studies, he has always been a great mentor for me. He set an example of excellence as a researcher, instructor and role model. My experiences in his group have always been rewarding. Conducting researches under his guidance motivates me to refine my skills and set a high standard for myself. I cherish this opportunity for professional and personal development that he has provided me.

I would like to thank my thesis committee members for providing me support through this process. Your suggestions and feedback have been invaluable for me.

I also want to thank our department for providing me with such an open and resourceful environment. I'm so glad that I will be able to continue enjoying my time here as a PhD student.

Last but not least, I can not tell how grateful I am to my parents. I would not have been here without your encouragement. You believe in me and always wanted the best for me. You are oceans away, yet your love is close.

# SAFETY-AWARE APPRENTICESHIP LEARNING

WEICHAO ZHOU

ABSTRACT

It is well acknowledged in the AI community that finding a good reward function for reinforcement learning is extremely challenging. Apprenticeship learning (AL) is a class of "learning from demonstration" techniques where the reward function of a Markov Decision Process (MDP) is unknown to the learning agent and the agent uses inverse reinforcement learning (IRL) methods to recover expert policy from a set of expert demonstrations. However, as the agent learns exclusively from observations, given a constraint on the probability of the agent running into unwanted situations, there is no verification, nor guarantee, for the learnt policy on the satisfaction of the restriction. In this dissertation, we study the problem of how to guid AL to learn a policy that is inherently safe while still meeting its learning objective. By combining formal methods with imitation learning, a Safety-Aware Apprenticeship Learning algorithm is proposed. We consider a setting where the unknown reward function is assumed to be a linear combination of a set of state features, and the safety property is specified in Probabilistic Computation Tree Logic (PCTL). By embedding probabilistic model checking inside AL, we propose a novel *counterexample-guided* approach that can ensure both safety and performance of the learnt policy. This algorithm guarantees that given some formal safety specification defined by probabilistic temporal logic, the learnt policy shall satisfy this specification. We demonstrate the effectiveness of our approach on several challenging AL scenarios where safety is essential.

# Contents

# List of Tables

# List of Figures

# List of Abbreviations

| | | |
|---|---|---|
| AI | ..... | Artificial Intelligence |
| AL | ..... | Apprenticeship Learning |
| IRL | ..... | Inverse Reinforcement Learning |
| Safe-AL | ..... | Safety-Aware Apprenticeship Learning |
| CEGIS | ..... | Counterexample-Guided Inductive Synthesis |
| CEX | ..... | Counterexample |
| PCTL | ..... | Probabilistic Computational Tree Logic |
| CEGAL | ..... | Counterexample-Guided Apprenticeship Learning |

# Chapter 1

# Introduction

**AI safety** has been one of the main factors limiting the deployment of artificial intelligence (Bundy, 2017). As AI technologies are gaining rapid progress in a diversity of areas, such as computer vision (Krizhevsky et al., 2012), video games (Mnih et al., 2015), and autonomous driving (Levinson et al., 2011), concerns about unintended consequences of widespread adoption have been raised. When exposed to the full complexity of the human environment, an AI agent should not only be able to finish the task assigned by human but also ensure that its probability of performing unsafe actions is at least below some acceptable threshold. This is especially significant in safety critical areas. As highlighted in (Amodei et al., 2016), if the objective function of an AI agent is wrongly specified, then maximizing that objective function may lead to harmful results. In addition, if an agent focuses only on accomplishing a specific task and ignore other aspects, such as safety constraints, of the environment, it may also perform unsafe behavior. But among various reasons why an agent learns an unsafe policy, the direct and intuitive cause is that the agent lacks for awareness of unsafe situations.

In this dissertation, we focus on the safety problems when AI learns from human demonstration and explore the following thesis:

*Given a safety specification, by learning from human demonstration with a verification-guided algorithm, an agent can ensure the safety of apprenticeship learning while retaining its performance.*

**Apprenticeship learning (AL)** approach developed by Abbeel and Ng (Abbeel and Ng, 2004) is one form of learning from demonstration where an agent tries to recover an expert's strategy by observing and learning from the expert's demonstration. This approach uses *inverse reinforcement learning* (Ng and Russell, 2000) technique in which it is assumed that expert makes decisions optimally in the environment. Apprenticeship Learning bypasses the explicit search of reward function as in reinforcement learning and directly recovers the expert's policy by matching the features of the expert's demonstration. However, the expert often can only demonstrate how the task works but not how the task may fail. This is because failure may cause irrecoverable damages to the system such as crashing a vehicle. In general, the lack of "negative examples" can cause a heavy bias in how the learning agent constructs the reward estimate. In fact, even if all the demonstrations are safe, the agent may still end up learning an unsafe policy.

**Probabilistic Model Checking** is a range of techniques calculating the likelihood of the occurrence of certain events during the execution of a probabilistic system (Kwiatkowska et al., 2002). Considering a safety specification expressed in probabilistic computation tree logic (PCTL) (Hansson and Jonsson, 1994), we employ a verification-in-the-loop approach by embedding probabilistic model checking (Kwiatkowska et al., 2002) as a safety checking mechanism inside the learning phase of AL. Inspired by the work on formal inductive synthesis (Jha and Seshia, 2017) and counterexample-guided inductive synthesis (Solar-Lezama et al., 2006), we incorporate formal verification in apprenticeship learning and propose a framework that uses the verification results to inductively improve the learnt policy. More specifically, when a learnt policy does not satisfy the PCTL formula, we leverage counterexamples generated by the model checker to steer the policy search in AL. In essence, counterexample generation can be viewed as supplementing negative examples for the learner. Thus, the learner will try to find a policy that not only imitates

the expert's demonstrations but also stays away from the failure scenarios as captured by the counterexamples.

## 1.1 Contributions

In summary, we make the following contributions in this thesis.

- We propose a novel framework for incorporating formal safety guarantees in learning from demonstrations.
- We develop a novel algorithm called CounterExample Guided Apprenticeship Learning (CEGAL) that combines probabilistic model checking with the optimization-based approach of apprenticeship learning.
- We demonstrate that the proposed approach can guarantee safety for a set of case studies and attain performance comparable to using apprenticeship learning alone.

## 1.2 Related Work

A taxonomy of AI safety problems is given in (Amodei et al., 2016) where the issues of misspecified objective or reward and insufficient or poorly curated training data were highlighted. There have been several efforts attempting to address these issues from different angles. The problem of *safe exploration* is studied in (Moldovan and Abbeel, 2012) and (Held et al., 2017). In particular, the latter work proposes to add a safety constraint on amount of damage, to the optimization problem so that the optimal policy can maximize the reward without violating the limit on the expected damage. An obvious shortcoming of this approach is that actual failures will have to occur so that the constraint can be properly decided.

Formal methods have been applied to the problem of AI safety. In (Gillulay and Tomlin, 2011), the authors propose to combine machine learning and reachability

analysis for dynamical models to achieve high performance and guarantee safety. In (Sadigh et al., 2014), the authors propose to use formal specification to synthesize a control policy for reinforcement learning. Recently, the problem of *safe reinforcement learning* was explored in (Alshiekh et al., 2017) where a monitor (called shield) is used to enforce temporal logic properties by providing a list of safe actions each time the agent makes a decision so that the temporal property is preserved. In (Junges et al., 2016), the authors also propose an approach for controller synthesis for reinforcement learning by using an SMT-solver is used to find a scheduler (policy) so that it satisfies both a probabilistic reachability property and an expected cost property. In (Mason et al., 2017), a so-called abstract Markov decision process (AMDP) model of the environment is first built and PCTL model checking is then used to check the satisfiability of safety specification. Our work is similar to these in spirit in the application of formal methods. However, while the concept of AL is closely related to reinforcement learning, an agent in the AL paradigm needs to learn a policy from demonstrations without knowing the reward function. A related safety problem in verification is the faithfulness of the model when it is learned from data. In (Puggelli et al., 2013), the authors propose a convex-MDP model for capturing uncertainties in the transition probabilities of an MDP as convex sets and propose a polynomial-time algorithm for verifying PCTL properties on these convex-MDPs.

Among imitation or apprenticeship learning methods, margin based algorithms (Abbeel and Ng, 2004), (Ng and Russell, 2000), (Ratliff et al., 2006) try to maximize the margin between the expert's policy and all learnt policies until the one with the smallest margin is produced. The apprenticeship learning algorithm in (Abbeel and Ng, 2004) was largely motivated by the support vector machine (SVM). Our algorithm makes use of this observation when using counterexamples to steer the policy search process. Recently, the idea of learning from failed demonstrations started to emerge. In (Shiarlis et al., 2016), the authors propose an IRL algorithm that can learn from

both successful and failed demonstrations. It is done by reformulating maximum entropy algorithm in (Ziebart et al., 2008) to find a policy that maximally deviates from the failed demonstrations while approaches the successful ones as much as possible. However, this entropy-based method requires obtaining many failed demonstrations and can be very costly in practice.

Finally, our approach is inspired by the work on formal inductive synthesis (Jha and Seshia, 2017) and counterexample-guided inductive synthesis (CEGIS) (Solar-Lezama et al., 2006). These frameworks typically combine a constraint-based synthesizer with a verification oracle. In each iteration, the agent refines her hypothesis (i.e. generates a new candidate solution) based on counterexamples provided by the oracle. Our approach can be viewed as an extension of CEGIS where the objective is not just functional correctness but also meeting certain learning criteria.

## 1.3 Outline

The rest of the thesis is organized as follows. Chapter 2 reviews background information on apprenticeship learning and PCTL model checking. Chapter 3.1 defines the safety-aware apprenticeship learning problem and gives an overview of our approach. Chapter 3.2 illustrates the counterexample-guided learning framework. Chapter 3.3 and 3.4 describes the proposed algorithm in detail. Chapter 4 presents a set of experimental results demonstrating the effectiveness of our approach.

# Chapter 2

# Background

## 2.1 Markov Decision Process

**Definition 2.1.1. Markov Decision Process (MDP)** is a tuple ($S$, $A$, $T$, $\gamma$, $s_0$, $R$), where $S$ is a finite set of states; $A$ is a set of actions; $T : S \times A \times S \to [0, 1]$ is a transition function describing the probability of transitioning from one state $s \in S$ to another $s' \in S$ by taking action $a \in A$ in state $s$; $R : S \to \mathbb{R}$ is a reward function which maps each state $s \in S$ to a real number indicating the reward of being in state $s$; $s_0 \in S$ is the initial state; $\gamma \in [0, 1)$ is a discount factor which describes the desirability of the future rewards.

A policy $\pi$ for an MDP $M$ induces a Discrete-Time Markov Chain (DTMC) $M_\pi = (S, T_\pi, s_0)$, where $T_\pi : S \times S \to [0, 1]$ is the probability of transitioning from a state $s$ to another state in one step. By making a sequence of decisions by following policy $\pi$, the agent shall traverse a trajectory $\tau = s_{t_0} \xrightarrow{T_\pi(s_{t_0}, s_{t_1}) > 0} s_{t_1} \xrightarrow{T_\pi(s_{t_1}, s_{t_2}) > 0} s_{t_2}, ...,$ is a sequence of states where $s_{t_i} \in S$. The accumulated reward of such trajectory $\tau$ is $\sum_{i=0}^{\infty} \gamma^i R(s_{t_i})$. The value function $V_\pi : S \to \mathbb{R}$ measures the expectation of accumulated reward $E[\sum_{i=0}^{\infty} \gamma^i R(s_{t_i})]$ starting from a state $s$ and following policy $\pi$.

According to (Bellman, 1957), a policy $\pi$ is optimal for MDP M if:

$$\forall s \in S, \pi(s) \in \underset{a \in A}{argmax} \sum_{s' \in S} T(s, a, s') V_\pi(s') \tag{2.1}$$

*Inverse reinforcement learning* (Ng and Russell, 2000) assumes that a learning

agent is provided with a set of m trajectories $E = \{\tau_0, \tau_1, ..., \tau_{m-1}\}$ from expert. The goal is to find a function $R$ that can generate optimal behavior similar to $E$.

*Apprenticeship learning* (Abbeel and Ng, 2004) aims at producing a policy which performs almost as well as the expert without guaranteeing to recover the reward function. It is assumed that reward function is the result of linear combination $R(s) = \omega^T f(s)$, where $f : S \rightarrow [0,\ 1]^k$ is a vector of feature functions on S and $\omega \in [0,1]^k \wedge ||\omega||_2 \leq 1$, is a weight vector. Feature function $f$ is known by both expert and the learning agent. Given some weight vector $\omega$, its corresponding reward function R is known and optimal policy $\pi$ can be solved. The expectation of values of the initial states distributed in D can be expressed as:

$$\begin{aligned} V_\pi(s_0) = E_\pi[\sum_{i=0}^{\infty} \gamma^t R(s_{t_i}) | s_{t_0} = s_0] &= E_\pi[\sum_{i=0}^{\infty} \gamma^t \omega^T f(s_{t_i}) | s_{t_0} = s_0] \\ &= \omega^T E_\pi[\sum_{i=0}^{\infty} \gamma^t f(s_{t_i}) | s_{t_0} = s_0] \end{aligned} \tag{2.2}$$

Define $\mu_\pi = E_\pi[\sum_{i=0}^{\infty} \gamma^t f(s_{t_i}) | s_{t_0} = s_0]$ as the expected features of policy $\pi$. Given a policy $\pi$, then $\mu_\pi$ can be solved through Monte Carlo method, or value iteration, or linear programming. If expert has a weight vector $\omega_E$ and a policy $\pi_E$, the expected features of expert's policy $\mu_E$ can be expressed in the same way. However, practically only a limited amount of demonstrations will be provided by the expert. Define $\hat{\mu}_E$ as the expected features of expert's m trajectories. If m is large enough, $\mu_E$ can be approximated by $\hat{\mu}_E$.

$$\begin{aligned} E[\sum_{i=1}^{i=m} \sum_{s_{t_j}^{(i)} \in \tau_i} \gamma^j R(s_{t_j}^{(i)})] = E[\sum_{i=1}^{i=m} \sum_{s_{t_j}^{(i)} \in \tau_i} \gamma^j \omega^T f(s_{t_j}^{(i)})] &= \omega^T E[\sum_{i=1}^{i=m} \sum_{s_{t_j}^{(i)} \in \tau_i} \gamma^j f(s_{t_j}^{(i)})] \\ &= \omega^T \hat{\mu}_E \end{aligned} \tag{2.3}$$

The algorithm proposed by Abbeel and Ng (Abbeel and Ng, 2004) starts with a random policy $\pi_0$ and its expected features $\mu_{\pi_0}$. Assuming that in iteration $i$, a

set of $i$ candidate policies $\Pi = \{\pi_0, \pi_1, ..., \pi_{i-1}\}$ and their corresponding expected features $\{\mu_\pi | \pi \in \Pi\}$ have been found, the algorithm solves the following optimization problem.

$$\delta = \max_{\omega} \min_{\pi \in \Pi} \omega^T (\hat{\mu}_E - \mu_\pi) \qquad s.t. \; ||\omega||_2 \leq 1 \tag{2.4}$$

The optimal $\omega$ is used to find the corresponding optimal policy $\pi_i$ and the expected features $\mu_{\pi_i}$. If $\delta \leq \epsilon$, then the algorithm terminates and $\pi_i$ is produced as the output. Otherwise, $\mu_{\pi_i}$ is added to the set of features for the candidate policy set $\Pi$ and the algorithm continues to the next iteration.

## 2.2 Probabilistic Model Checking

Probabilistic model checking can be used to verify properties of a stochastic system such as "is the probability that the agent reaches the unsafe area within 10 steps smaller than 5%?". *Probabilistic Computation Tree Logic* (PCTL) (Hansson and Jonsson, 1994) allows for probabilistic quantification of those properties. The syntax of PCTL includes state formulas and path formulas (Kwiatkowska et al., 2002). A state formula $\phi$ asserts property of a single state $s \in S$ whereas a path formula $\psi$ asserts property of a trajectory.

$$\phi \quad ::= \quad true \mid l \mid \neg\phi_i \mid \phi_i \wedge \phi_j \mid P_{\bowtie p^*}[\psi] \tag{2.5}$$

$$\psi \quad ::= \quad \mathbf{X}\phi \mid \phi_1 \mathbf{U}^{\leq k} \phi_2 \mid \phi_1 \mathbf{U} \phi_2 \tag{2.6}$$

where $l \in AP$ is an atomic proposition; $\bowtie \in \{\leq, \geq, <, >\}$; $P_{\bowtie p^*}[\psi]$ means that the probability of generating a trajectory that satisfies formula $\psi$ is $\bowtie p^*$. $\mathbf{X}\phi$ asserts that the next state after initial state in the trajectory satisfies $\phi$; $\phi_1 \mathbf{U}^{\leq k} \phi_2$ asserts that $\phi_2$ is satisfied in at most $k$ transitions and all preceding states satisfy $\phi_1$; $\phi_1 \mathbf{U} \phi_2$ asserts that $\phi_2$ will be eventually satisfied and all preceding states satisfy $\phi_1$.

The semantics of PCTL is defined by a satisfaction relation $\models$ between formula

and DTMC:

$$s \models true \text{ iff state } s \in S \tag{2.7}$$

$$s \models a \quad \text{iff } a \in L(s) \tag{2.8}$$

$$s \models \phi \quad \text{iff state s satisfies state formula } \phi \tag{2.9}$$

$$s \models \neg\phi \quad \text{iff } s| = \phi \text{ is false} \tag{2.10}$$

$$s \models \phi_1 \wedge \phi_2 \quad \text{iff } s \models \phi_1 \text{ and } s \models \phi_2 \tag{2.11}$$

$$s \models P_{\bowtie p^*}(\psi) \quad \text{iff } Prob(s, \psi) \bowtie p^* \tag{2.12}$$

The satisfaction relation $\models$ between trajectory $\tau$ and path formula $\psi$ is defined as below:

$$\tau \models \psi \quad \text{iff } \tau \text{ satisfies } \psi \tag{2.13}$$

$$\tau \models \phi_1 U^{\leq k} \phi_2 \quad \text{iff } \exists j \leq k, \tau(s_{t_j}) \models \phi_1 \wedge \forall 0 \leq i \leq j, \tau(s_{t_i}) \models \psi \tag{2.14}$$

$$\tau \models \phi_1 W^{\leq k} \phi_2 \quad \text{iff } \tau \models \phi_1 U^{\leq k} \phi_2 \text{ or } \forall i \leq k, s_{t_i} \models \phi_1) \tag{2.15}$$

In this thesis a third-party probabilistic model checking tool, PRISM (Kwiatkowska et al., 2002), is used. It can take a description of a system, which is a DTMC in this dissertation, with a set of PCTL specifications as inputs, then answers which states of the system satisfy each specification. This is a symbolic model checker (Fujita et al., 1997) which uses binary decision diagram (BDD) and multi-terminal BDDs (MTBDDs) as the underlying data structures. The tool reduces the verification problem to reachability-based computation and numerical calculation. The data structures can help the tool perform efficient and fast computation on large, structured models (Kwiatkowska et al., 2002).

A policy can also be synthesized by using model checking tool to solve the objective $\min_{\pi} P_{=?}[\psi]$ for an MDP $M$. Conversely, the solved policy can be used to verify

the satisfiability of $P_{\leq p^*}[\psi]$. This problem can be solved by linear programming or policy iteration (and value iteration for step-bounded reachability) (Kwiatkowska and Parker, 2013).

## 2.3 Counterexample

One major strength of probabilistic model checking techniques is to generate counterexamples in case a temporal logic property is violated (Han et al., 2009). Counterexample can be viewed as a proof of the violation. In this thesis, it is used as demonstrations of unsafe behaviors.

In addition to the satisfaction relations in PCTL semantics, $\models_{min}$ denotes the minimal satisfaction relation (Han et al., 2009) between $\tau$ and $\psi$. Defining $pref(\tau)$ as the set of all prefixes of trajectory $\tau$ including $\tau$ itself, then $\tau \models_{min} \psi$ iff $(\tau \models \psi) \wedge (\forall \tau' \in pref(\tau) \backslash \tau, \tau' \not\models \psi)$. For instance, if $\psi = \phi_1 \mathbf{U}^{\leq k} \phi_2$, then for any finite trajectory $\tau \models_{min} \phi_1 \mathbf{U}^{\leq k} \phi_2$, only the final state in $\tau$ satisfies $\phi_2$. Let $P(\tau)$ be the probability of transitioning along a trajectory $\tau$ and let $\Gamma_\psi$ be the set of all finite trajectories that satisfy $\tau \models_{min} \psi$, the value of PCTL property $\psi$ is defined as $P_{=?|s_0}[\psi] = \sum_{\tau \in \Gamma_\psi} P(\tau)$. For a DTMC $M_\pi$ and a state formula $\phi = P_{\leq p^*}[\psi]$, $M_\pi \models \phi$ iff $P_{=?|s_0}[\psi] \leq p^*$.

There can be different counterexamples for one formula. Let $\mathbb{P}(\Gamma) = \sum_{\tau \in \Gamma} P(\tau) \leq p^*$ be the sum of probabilities of all trajectories in one set. Assumes that all counterexamples for formula $\phi$ are gathered in a set $CEX_\phi \subset 2^{\Gamma_\psi}$ such that $(\forall CEX \in CEX_\phi, \mathbb{P}(CEX) \geq p^*) \wedge (\forall \Gamma \in 2^{\Gamma_\psi} \backslash CEX_\phi, \mathbb{P}(\Gamma) < p^*)$. The following definitions can be found in (Han et al., 2009).

**Definition 2.3.1. Minimal counterexample** is a $CEX \in CEX_\phi$ which satisfies that $\forall CEX' \in CEX_\phi, |CEX| \leq |CEX'|$. There can be multiple minimal counterexamples in $CEX_\phi$.

**Definition 2.3.2. Smallest counterexample** is a minimal counterexample $CEX \in$

$CEX_\phi$ which additionlly satisfies $\mathbb{P}(CEX) \leq \mathbb{P}(CEX')$ for any other minimal counterexample $CEX' \in CEX_\phi$.

By converting DTMC $M_\pi$ into a weighted directed graph, counterexample can be found by solving a k-shortest paths (KSP) problem or a hop-constrained KSP (HKSP) problem (Han et al., 2009). Alternatively, counterexamples can be found by using Satisfiability Modulo Theory solving or mixed integer linear programming to determine the minimal critical subsystems that capture the counterexamples in $M_\pi$ (Wimmer et al., 2012).

In this thesis a third-party tool, COMICS (Jansen et al., 2012), is used to generate minimal counterexample for a DTMC. This tool performs SCC-based model checking (Ábrahám et al., 2010) to a DTMC and a PCTL property. If model checking results refutes the PCTL property, a counterexample can be computed and represented either as a set of paths or as a critical subsystem. In this dissertation, we use the set of paths representation.

# Chapter 3

# Counterexample-Guided Apprenticeship Learning

## 3.1 Motivating example

We will first analyze the safety issues in apprenticeship learning with a grid-world example. Then we will define the safety-aware apprenticeship learning (SafeAL) problem and give intuitions on how we solve it.

Assuming that there are some unsafe states in an $MDP\backslash R$ $M = (S, A, T, \gamma, s_0)$. A safety issue means an agent following a learnt policy has a higher probability of entering those unsafe states than it should. There are multiple reasons that can give rise to this issue. First, it is possible that the expert policy itself has a high probability of reaching the unsafe states. Second, human experts often tend to perform only successful demonstrations that do not highlight the unwanted situations (Shiarlis et al., 2016). This *lack of negative examples* in the training set results in the agent being unaware of the existence of those unsafe states.

We use a 8 x 8 grid world navigation example as shown in Fig. 3·1 to illustrate this problem. The agent starts from the upper-left corner and moves from cell to cell until reaching the lower-right corner or a maximal step length $t < 64$. Meanwhile, there are several cells labelled as unsafe enclosed by the red dashed lines shown near the upper-right and lower-left corners. These are regions that agent should avoid. In each time step, the agent can choose to stay in current cell or move to an adjacent cell.

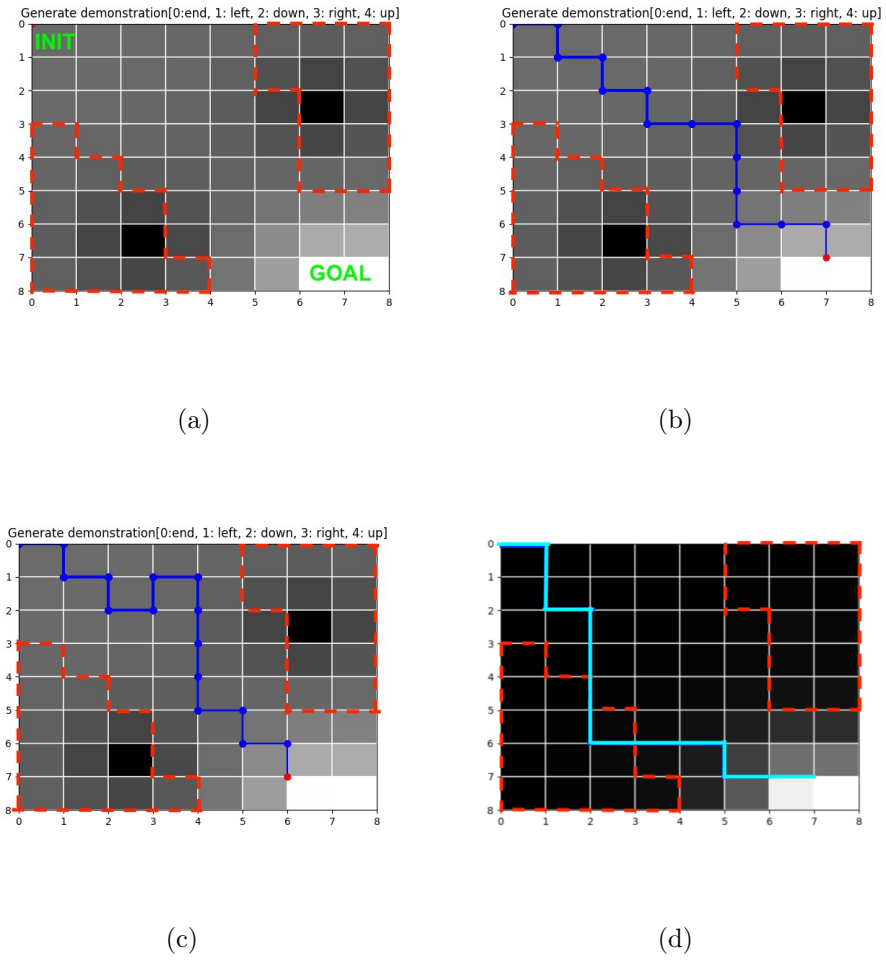



**Figure 3·1:** (a) The 8 x 8 gridworld. Lighter grid cells indicate relatively higher rewards while darker ones indicate lower rewards. It is regarded by the expert that the two black grid cells have the lowest rewards, while the two white ones constituting the goal area have the the highest rewards. The grid cells enclosed by red dashed lines are labelled as unsafe. (b) The blue line is the first representative trajectory that expert perform during demonstration. (c) The blue line is the second representative trajectory that expert perform during demonstration. (d) The rewards of the goal states have very high contrast against all other states. The difference between unsafe states and nearby states is so small that the agent has a high probability of performing a trajectory passing through the unsafe states as indicated by the cyan line.

Due to stochasticity of the system, it has 30% chance of moving randomly instead of moving accroding its decision. The expert knows the goal area, the unsafe area

and the reward mapping on all states as shown in Fig. 3·1(a). For each state $s \in S$, the feature vector $f(s)$ consists of 4 feature functions $f_i(s)$, $i = 0, 1, 2, 3$. All of them are radial basis functions which respectively depend on the squared Euclidean distances between $s$ and the 4 states with the highest or lowest rewards as shown in Fig. 3·1(a). In addition, a specification $\Phi$ formalized in PCTL is given to capture the safety requirement. In Eq. 3.1, $p^*$ is the upper bound of the probability of reaching an unsafe state within $t = 64$ steps and can be set by any value in $[0, 1]$ initially.

$$\Phi ::= P_{=?|s_0}[\texttt{true}\ \mathbf{U}^{\leq t}\ \texttt{unsafe}] \leq p^* \tag{3.1}$$

We extend the satisfaction relation $\models$ in PCTL and define that a policy $\pi \models \Phi$ if the initial state $s_0$ of the DTMC $M_\pi$ induced from MDP $M$ by policy $\pi$ satisfies the PCTL property in $\Phi$. We illustrate two scenarios in this simple example. The first simulates a setting with abundant expert demonstrations, i.e. $\mu_E$ is directly generated from the optimal policy $\pi_E$ with respect to the predetermined $\omega_E$ which results in the reward mapping in Fig. 3·1(a). In this case, the AL algorithm can accurately recover $\pi_E$. Model checking result shows that the probability of reaching an unsafe state by following the learnt policy, or the expert policy $\pi_E$ itself, is 11.7%. Hence, the specification is satisfied in this scenario. In the second scenario, which is more realistic, the expert follows $\pi_E$ but only performs a limited number of demonstrations which are all successful and safe. As indicated by the two representative (in blue) trajectories shown in Fig. 3·1(b) and Fig. 3·1(c), $10,000$[1] trajectories were used as expert demonstrations. The reward function that induces the learnt policy in this scenario is shown in Fig. 3·1(d). Observe that only the goal area has been learnt whereas the agent is oblivious to the unsafe regions (treating them in the same way as other black cells). Indeed, probability of reaching an unsafe state within 64 steps with this policy turns out to be 98% (thus violating the safety requirement by a large

[1]This number is actually considered small for the size of this problem.

margin). To make matters worse, the value of $p^*$ may be decided or revised after a policy has been learnt. In that case, even the original expert policy $\pi_E$ is unsafe, e.g., when $p^* = 10\%$. Thus, we need to adapt the AL algorithm to account for this additional safety requirement.

**Definition 3.1.1. The safety-aware apprenticeship learning (SafeAL) problem** is, given an $MDP \backslash R$, a set of $m$ trajectories $\{\tau_0, \tau_1, ..., \tau_{m-1}\}$ demonstrated by an expert, and a specification $\Phi$, to learn a policy $\pi$ that satisfies $\Phi$ and is $\epsilon$-close to the expert policy $\pi_E$.

## 3.2 A Framework for Safety-Aware Learning

In this section, we describe a general framework for safety-aware learning. This novel framework utilizes information from both the expert demonstrations and a verifier. The proposed framework is illustrated in Fig. 3·2. Similar to the *counterexample-guided inductive synthesis* (CEGIS) paradigm (Solar-Lezama et al., 2006), our framework consists of a *verifier* and a *learner*. The verifier checks if a candidate policy satisfies the safety specification $\Phi$. In case $\Phi$ is not satisfied, the verifier generates a counterexample for $\Phi$. The main difference from CEGIS is that our framework considers not only functional correctness, e.g., safety, but also performance (as captured by the learning objective). Starting from an initial policy $\pi_0$, each time the learner learns a new policy, the verifier checks if the specification is satisfied. If true, then this policy is added to the candidate set, otherwise the verifier will generate a (minimal) counterexample and add it to the counterexample set. During the learning phase, the learner uses both the counterexample set and candidate set to find a policy that is close to the (unknown) expert policy and far away from the counterexamples. The goal is to find a policy that is $\epsilon$-close to the expert policy and satisfies the specification.

Learning from a (minimal) counterexample $cex_\pi$ of a policy $\pi$ is similar to learning from expert demonstrations. The basic principle of the AL algorithm proposed in

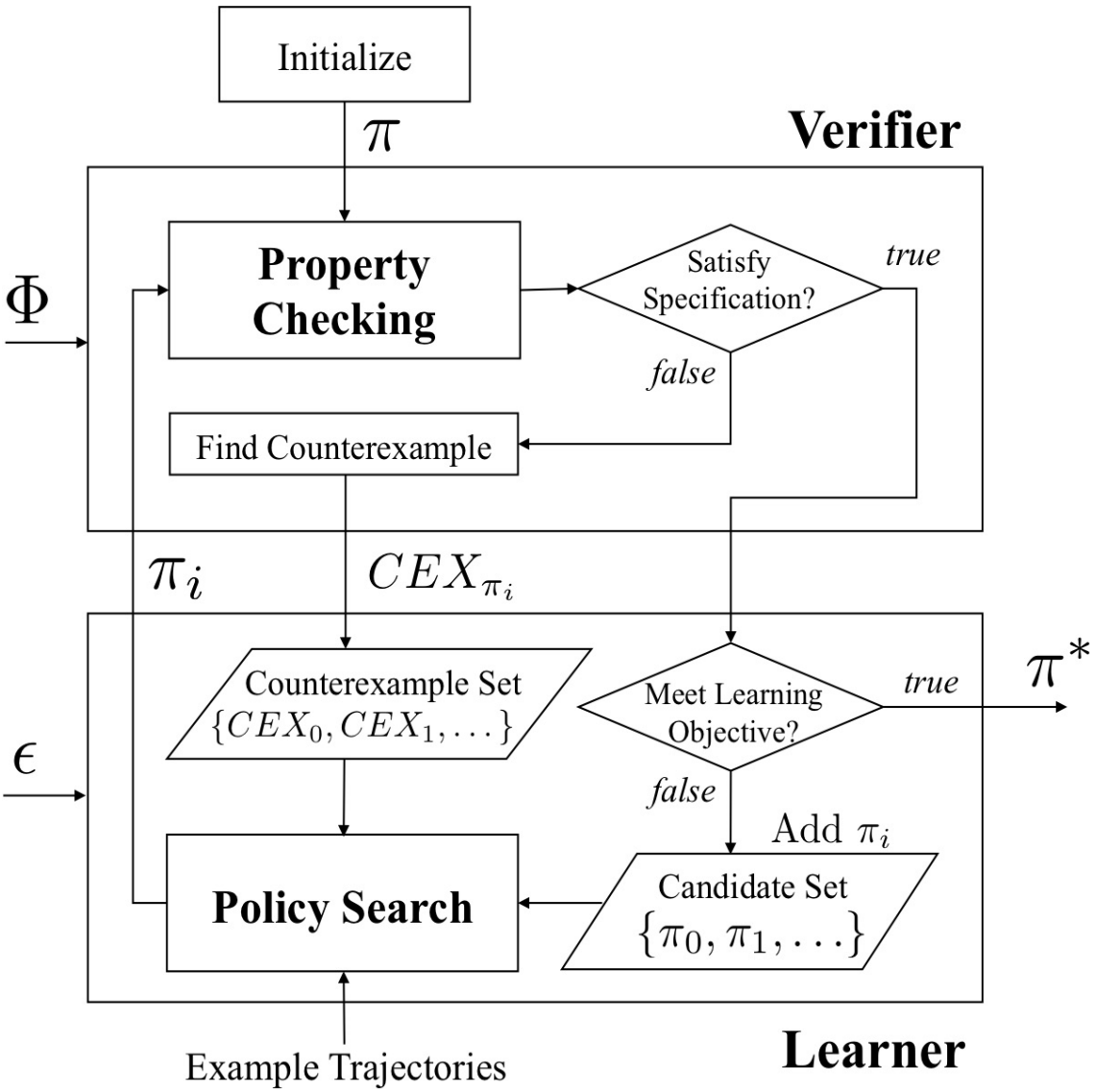
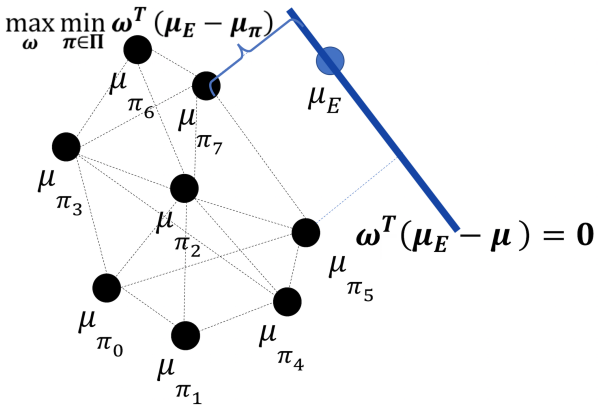
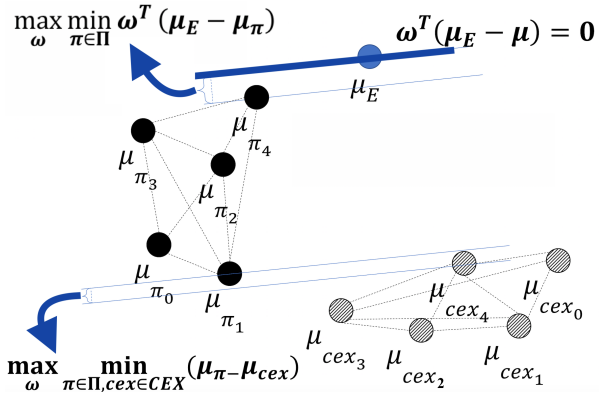



**Figure 3·2:** Our safety-aware learning framework. Given an initial policy $\pi_0$, a specification $\Phi$ and a learning objective (as captured by $\epsilon$), the framework iterates between a *verifier* and a *learner* to search for a policy $\pi^*$ that satisfies both $\Phi$ and $\epsilon$. One invariant that this framework maintains is that all the $\pi_i$'s in the candidate policy set satisfy $\Phi$.




(a)




(b)

**Figure 3·3:** (a) Learn from expert. (b) Learn from both expert demonstrations and counterexamples.

(Abbeel and Ng, 2004) is to find a weight vector $\omega$ under which the expected reward of $\pi_E$ maximally outperforms any mixture of the policies in the candidate policy set $\Pi = \{\pi_0, \pi_1, \pi_2, \ldots\}$. Thus, $\omega$ can be viewed as the normal vector of the hyperplane $\omega^T(\mu - \mu_E) = 0$ that has the maximal distance to the convex hull of the set $\{\mu_\pi | \pi \in \Pi\}$ as illustrated in the 2D feature space in Fig. 3·3(a). It can be shown that $\omega^T \mu_\pi \geq \omega^T \mu_{\pi'}$ for all previously found $\pi'$s. Intuitively, this moves the candidate $\mu_\pi$ closer to $\mu_E$. Similarly, we can apply the same max-margin separation principle to maximize the distance between the candidate policies and the counterexamples (in the $\mu$ space).

Let $CEX = \{cex_0, cex_1, cex_2, ...\}$ denote the set of counterexamples of the policies that do not satisfy the specification $\Phi$. Maximizing the distance between the convex hulls of the sets $\{\mu_{cex} \mid cex \in CEX\}$ and $\{\mu_\pi \mid \pi \in \Pi\}$ is equivalent to maximizing the distance between the parallel supporting hyperplanes of the two convex hulls as shown in Fig. 3·3(b). The corresponding optimization function is given in Eq. (3.2).

$$\delta = \max_{\omega} \min_{\pi \in \Pi, cex \in CEX} \omega^T(\mu_\pi - \mu_{cex}) \qquad s.t. \; ||\omega||_2 \leq 1 \tag{3.2}$$

To attain good performance similar to that of the expert, agent still needs to learn from $\mu_E$. Thus, the overall problem can be formulated as a multi-objective optimization problem that combines (2.4) and (3.2) into (3.3).

$$\max_{\omega} \min_{\pi \in \Pi, \tilde{\pi} \in \Pi, cex \in CEX} (\omega^T(\mu_E - \mu_\pi), \; \omega^T(\mu_{\tilde{\pi}} - \mu_{cex})) \qquad s.t. \; ||\omega||_2 \leq 1 \tag{3.3}$$

## 3.3 An Algorithm for Counterexample-Guided Apprenticeship Learning

In this section, we introduce the Counterexample Guided Apprenticeship Learning (CEGAL) algorithm to solve the Safe-AL problem. It can be viewed as a special case of the safety-aware learning framework described in the previous section. In addition to combining policy verification, counterexample generation and AL, our approach uses an adaptive weighting scheme to weight the separation from $\mu_E$ with the separation from $\mu_{cex}$.

$$\begin{aligned} &\max_{\omega} \min_{\pi \in \Pi_S, \tilde{\pi} \in \Pi_S, cex \in CEX} \omega^T(k(\mu_E - \mu_\pi) + (1-k)(\mu_{\tilde{\pi}} - \mu_{cex})) \\ &s.t. \; ||\omega||_2 \leq 1, \; k \in [0,1] \\ &\quad \omega^T(\mu_E - \mu_\pi) \leq \omega^T(\mu_E - \mu_{\pi'}), \; \forall \pi' \in \Pi_S \\ &\quad \omega^T(\mu_{\tilde{\pi}} - \mu_{cex}) \leq \omega^T(\mu_{\tilde{\pi}'} - \mu_{cex'}), \; \forall \tilde{\pi}' \in \Pi_S, \forall cex' \in CEX \end{aligned} \tag{3.4}$$

**Algorithm 1** Counterexample-Guided Apprenticeship Learning (CEGAL)

**Input**:
  $M \leftarrow$ A partially known $MDP \backslash R$; $f \leftarrow$ A vector of feature functions
  $\mu_E \leftarrow$ The expected features of expert trajectories $\{\tau_0, \tau_1, \ldots, \tau_m\}$
  $\Phi \leftarrow$ Specification; $\epsilon \leftarrow$ Error bound for the expected features;
  $\sigma, \alpha \in (0, 1) \leftarrow$ Error bound $\sigma$ and step length $\alpha$ for the parameter $k$;
**Initialization**:
  **If** $||\mu_E - \mu_{\pi_0}||_2 \leq \epsilon$, **then return** $\pi_0$ ▷ $\pi_0$ is the **initial safe policy**
  $\Pi_S \leftarrow \{\pi_0\}$, $CEX \leftarrow \emptyset$ ▷ Initialize candidate and counterexample set
  $inf \leftarrow 0, sup \leftarrow 1, k \leftarrow sup$ ▷ Initialize multi-optimization parameter $k$
  $\pi_1 \leftarrow$ Policy learnt from $\mu_E$ via apprenticeship learning
**Iteration** $i$ $(i \geq 1)$:
  **Verifier:**
    status $\leftarrow Model\_Checker(M, \pi_i, \Phi)$
    **If** status = SAT, **then go to Learner**
    **If** status = UNSAT
      $cex_{\pi_i} \leftarrow Counterexample\_Generator(M, \pi_i, \Phi)$
      Add $cex_{\pi_i}$ to $CEX$ and solve $\mu_{cex_{\pi_i}}$, **go to Learner**
  **Learner:**
    **If** status = SAT
      **If** $||\mu_E - \mu_{\pi_i}||_2 \leq \epsilon$, **then return** $\pi_i$
        ▷ Converge. $\pi_i$ is $\epsilon$-close to $\pi_E$
      Add $\pi_i$ to $\Pi_S$, $inf \leftarrow k, k \leftarrow sup$ ▷ Update $\Pi_S$, $inf$ and reset $k$
    **If** status = UNSAT
      **If** $|k - inf| \leq \sigma$, **then return** $\pi^* \leftarrow \underset{\pi \in \Pi_S}{argmin} ||\mu_E - \mu_\pi||_2$
        ▷ Converge. $k$ is too close to its lower bound.
      $k \leftarrow \alpha \cdot inf + (1 - \alpha)k$ ▷ Decrease k to learn for safety
    $\omega_{i+1} \leftarrow \underset{\omega}{argmax} \min_{\pi \in \Pi_S, \tilde{\pi} \in \Pi_S, cex \in CEX} \omega^T (k(\mu_E - \mu_\pi) + (1 - k)(\mu_{\tilde{\pi}} - \mu_{cex}))$
  ▷ Note that the multi-objective optimization function recovers AL when $k = 1$
    $\pi_{i+1}, \mu_{\pi_{i+1}} \leftarrow$ Compute the optimal policy $\pi_{i+1}$ and its expected features $\mu_{\pi_{i+1}}$ for the MDP $M$ with reward $R(s) = \omega_{i+1}^T f(s)$
  **Go to next iteration**

In essence, we take a weighted-sum approach for solving the multi-objective optimization problem (3.3). Assuming that $\Pi_S = \{\pi_1, \pi_2, \pi_3, \ldots\}$ is a set of candidate policies that all satisfy $\Phi$, $CEX = \{cex_1, cex_2, cex_3, \ldots\}$ is a set of counterexamples. We introduce a parameter $k$ and change (3.3) into a weighted sum optimization problem (3.4). Note that $\pi$ and $\tilde{\pi}$ can be different. The optimal $\omega$ solved from (3.4) can be used to generate a new policy $\pi_\omega$ by using algorithms such as policy iteration. We use

a probabilistic model checker, such as PRISM (Kwiatkowska et al., 2002), to check if $\pi_\omega$ satisfies $\Phi$. If it does, then it will be added to $\Pi_S$. Otherwise, a counterexample generator, such as COMICS (Jansen et al., 2012), is used to generate a (minimal) counterexample $cex_{\pi_\omega}$, which will be added to $CEX$.

Algorithm 1 describes CEGAL in detail. With a constant $sup = 1$ and a variable $inf \in [0, sup]$ for the upper and lower bounds respectively, the learner determines the value of $k$ within $[inf, sup]$ in each iteration depending on the outcome of the verifier and uses $k$ in solving (3.4) in line 27. Like most nonlinear optimization algorithms, this algorithm requires an initial guess, which is an initial safe policy $\pi_0$ to make $\Pi_S$ nonempty. A good initial candidate would be the maximally safe policy for example obtained using PRISM-games (Kwiatkowska et al., 2017). Without loss of generality, we assume this policy satisfies $\Phi$. Suppose in iteration $i$, an intermediate policy $\pi_i$ learnt by the learner in iteration $i-1$ is verified to satisfy $\Phi$, then we increase $inf$ to $inf = k$ and reset $k$ to $k = sup$ as shown in line 22. If $\pi_i$ does not satisfy $\Phi$, then we reduce $k$ to $k = \alpha \cdot inf + (1-\alpha)k$ as shown in line 26 where $\alpha \in (0, 1)$ is a step length parameter. If $|k - inf| \leq \sigma$ and $\pi_i$ still does not satisfy $\Phi$, the algorithm chooses from $\Pi_S$ a best safe policy $\pi^*$ which has the smallest margin to $\pi_E$ as shown in line 24. If $\pi_i$ satisfies $\Phi$ and is $\epsilon$-close to $\pi_E$, the algorithm outputs $\pi_i$ as show in line 19. For the occasions when $\pi_i$ satisfies $\Phi$ and $inf = sup = k = 1$, solving (3.4) is equivalent to solving (2.4) as in the original AL algorithm.

*Remark.* The initial policy $\pi_0$ does not have to be maximally safe, although such a policy can be used to verify if $\Phi$ is satisfiable at all. Naively safe policies often suffice for obtaining a safe and performant output at the end. Such a policy can be obtained easily in many settings, e.g., in the grid-world example one safe policy is simply staying in the initial cell. In both cases, $\pi_0$ typically has very low performance since satisfying $\Phi$ is the only requirement for it. An initial safe policy can also be generated (with no guarantee) by solving (3.2) iteratively as shown in Algorithm 2.

**Algorithm 2** Initial Safe Policy Generation

**Initialize**:
    $\pi_0 \leftarrow$Any arbitrary policy ▷ Can be any policy, e.g. the policy learnt via AL
    $\epsilon \leftarrow$ Error bound for the expected features
    $CEX \leftarrow \emptyset$
**Iteration**:
    In iteration $i \geq 1$, get $\pi_i$ from last iteration
    **Verifier:**
        status $\leftarrow Model\_Checker(M, \pi_i, \Phi)$
        **If** status = SAT, **then go to Learner**
        **If** status = UNSAT
            $cex_{\pi_i} \leftarrow Counterexample\_Generator(M, \pi_i, \Phi)$
            Add $cex_{\pi_i}$ to $CEX$ and solve $\mu_{cex_{\pi_i}}$, **go to Learner**
    **Learner:**
        **If** status = SAT
            Return $\pi_i$ ▷ End iteration once a safe policy is found
        **If** status = UNSAT
            **If** $|CEX| > 1$ and $\|\frac{\sum_{cex \in CEX} \mu_{cex}}{|CEX|} - \frac{\mu_{cex_i} + \sum_{cex \in CEX} \mu_{cex}}{1+|CEX|}\|_2 \leq \epsilon$
                **Then** return Null
    ▷ Fail to find a safe policy since adding a new counterexample hardly changes the centroid of existing counterexample cluster.
        $\omega_{i+1} = \underset{\omega}{argmax} \underset{cex \in CEX}{min} (-\omega^T \mu_{cex})$
        $\pi_{i+1} \leftarrow$ Optimal policy of $R(s) = \omega_{i+1}^T f(s)$
    Go to next iteration

**Theorem 3.3.1.** Given an initial policy $\pi_0$ that satisfies $\Phi$, Algorithm 1 is guaranteed to output a policy $\pi^*$, such that (1) $\pi^*$ satisfies $\Phi$, and (2) the performance of $\pi^*$ is at least as good as that of $\pi_0$ when compared to $\pi_E$, i.e. $\|\mu_E - \mu_{\pi^*}\|_2 \leq \|\mu_E - \mu_{\pi_0}\|_2$.

*Proof sketch.* The first part of the guarantee can be proven by case splitting. Algorithm 1 outputs $\pi^*$ either when $\pi^*$ satisfies $\Phi$ and is $\epsilon$-close to $\pi_E$, or when $|k - inf| \leq \sigma$ in some iteration. In the first case, $\pi^*$ clearly satisfies $\Phi$. In the second case,

$$(\pi^* = \underset{\pi \in \Pi_S}{argmin} ||\mu_E - \mu_\pi||_2) \wedge (\pi \models \Phi, \ \forall \pi \in \Pi_S) \tag{3.5}$$

is always true. Hence $\pi^*$ always satisfies $\Phi$. For the second part of the guarantee, the

initial policy $\pi_0$ itself is the final output if

$$(\pi_0 \models \Phi) \wedge (||\mu_E - \mu_{\pi_0}||_2 \leq \epsilon). \tag{3.6}$$

Otherwise, $\pi_0$ is added to $\Pi_S$ if it satisfies $\Phi$. During the iteration, since a safe policy $\epsilon$-close to $\mu_E$ will not be added to $\Pi_S$ but directly be returned, then

$$(||\mu_E - \mu_\pi||_2 \geq \epsilon, \forall \pi \in \Pi_S) \wedge \big((\pi^* = \underset{\pi \in \Pi_S}{argmin} ||\mu_E - \mu_\pi||_2)\big) \vee (||\mu_E - \mu_{\pi^*}||_2 \leq \epsilon)\big) \tag{3.7}$$

is always true. Hence $\|\mu_E - \mu_{\pi^*}\|_2 \leq \|\mu_E - \mu_{\pi_0}\|_2$ is always true.

*Discussion.* In the worst case CEGAL will return the initial safe policy. However, this can be because a policy that simultaneously satisfies $\Phi$ and is $\epsilon$-close to the expert's demonstrations does not exist. Comparing to AL which offers no safety guarantee and finding the maximally safe policy which has very poor performance, CEGAL provides a principled way of guaranteeing safety while retaining performance.

## 3.4 Convergence Analysis

Algorithm 1 is guaranteed to converge. Let $inf_t$ be the $t^{th}$ assigned value of $inf$. After $inf_t$ is given, $k$ is decreased from $k_0 = sup$ iteratively by $k_i = \alpha \cdot inf_t + (1-\alpha)k_{i-1}$ until either $|k_i - inf_t| \leq \sigma$ in line 24 or a new safe policy is found in line 18. The update of $k$ satisfies the following equality.

$$\frac{|k_{i+1} - inf_t|}{|k_i - inf_t|} = \frac{\alpha \cdot inf_t + (1-\alpha)k_i - inf_t}{k_i - inf_t} = 1 - \alpha \tag{3.8}$$

Thus, it takes no more than $1+\log_{1-\alpha} \frac{\sigma}{sup - inf_t}$ iterations for either the algorithm to converge in line 24 or a new safe policy to be found in line 18. If a new safe policy is found in line 18, $inf$ will be assigned in line 22 by the current value of $k$ as $inf_{t+1} = k$

which obviously satisfies $inf_{t+1} - inf_t \geq (1-\alpha)\sigma$. After the assignment of $inf_{t+1}$, the iterative update of $k$ resumes. Since $sup - inf_t \leq 1$, the following inequality holds.

$$\frac{|inf_{t+1} - sup|}{|inf_t - sup|} \leq \frac{sup - inf_t - (1-\alpha)\sigma}{sup - inf_t} \leq 1 - (1-\alpha)\sigma \tag{3.9}$$

Obviously, starting from an initial $inf = inf_0 < sup$, with the alternating update of $inf$ and $k$, $inf$ will keep getting close to $sup$ unless the algorithm converges as in line 24 or a safe policy $\epsilon$-close to $\pi_E$ is found as in line 19. The extreme case is that finally $inf = sup$ after no more than $\frac{sup - inf_0}{(1-\alpha)\sigma}$ updates on $inf$. Then, the problem becomes AL. Therefore, the worst case of this algorithm can have two phases. In the first phase, $inf$ increases from $inf = 0$ to $inf = sup$. Between each two consecutive updates $(t, t+1)$ on $inf$, there are no more than $\log_{1-\alpha} \frac{(1-\alpha)\sigma}{sup - inf_t}$ updates on $k$ before $inf$ is increased from $inf_t$ to $inf_{t+1}$. Overall, this phase takes no more than

$$\sum_{0 \leq i < \frac{sup - inf_0}{(1-\alpha)\sigma}} \log_{1-\alpha} \frac{(1-\alpha)\sigma}{sup - inf_0 - i \cdot (1-\alpha)\sigma} = \sum_{0 \leq i < \frac{1}{(1-\alpha)\sigma}} \log_{1-\alpha} \frac{(1-\alpha)\sigma}{1 - i \cdot (1-\alpha)\sigma} \tag{3.10}$$

iterations to reduce the multi-objective optimization problem to original apprenticeship learning and then the second phase begins. Since $k = sup$, the iteration will stop immediately when an unsafe policy is learnt as in line 24. This phase will not take more iterations than original AL algorithm does to converge.

The convergence analysis of AL can be found in (Abbeel and Ng, 2004). First it is assumed that the ball with radius $\epsilon$ around $\mu_E$, which is a set $\{\mu \mid d(\mu_E, \mu) \leq \epsilon\}$, will have intersection with be the convex hull of the set of feature expectations of candidate policy set $\Pi$ after $n$ iterations. Then it is provable that $n$ satisfies

$$n \leq O\Big(\frac{k}{(1-\gamma)^2\epsilon^2} \log \frac{k}{(1-\gamma)\epsilon}\Big) \tag{3.11}$$

where $k$ is the number of features, $\gamma$ is the discount factor. It is also ensured in (Abbeel

and Ng, 2004) that the number of sample demonstrations needed to make AL algorithm output with probability at least $1-\delta$ should suffice that

$$m \geq \frac{2k}{(1-\gamma)^2\epsilon^2} \log \frac{2k}{\delta} \tag{3.12}$$

In each iteration of Algorithm 1, the algorithm first solves a second-order cone programming (SOCP) problem (3.4) to learn a policy. SOCP problems can be solved in polynomial time by interior-point (IP) methods (Kuo and Mittelmann, 2004). PCTL model checking for DTMCs can be solved in time linear in the size of the formula and polynomial in the size of the state space (Hansson and Jonsson, 1994). Counterexample generation can be done either by enumerating paths using the $k$-shortest path algorithm or determining a critical subsystem using either a $SMT$ formulation or mixed integer linear programming (MILP) (Wimmer et al., 2012). For the $k$-shortest path-based algorithm, it can be computationally expensive sometimes to enumerate a large amount of paths (i.e. a large $k$) when $p^*$ is large. This can be alleviated by using a smaller $p^*$ during calculation, which is equivalent to considering only paths that have high probabilities.

## 3.5 Summary and Discussion

In this chapter we have posed a SafeAL problem in 3.1 with a motivation example and proposed to use max-margin separation principle to address it. The algorithm discussed in 3.2 formulates the max-margin separation principle with a multi-objective optimization problem and solves it with a weight-sum approach. The algorithm resembles AL algorithm from (Abbeel and Ng, 2004) in generating candidate policies iteratively. However, given a satisfiable safety specification described in PCTL, our algorithm provides guarantee of the safety of the output policy. We also analyzed the the performance guarantee of the final output as well as the convergence of the

iteration.

Nonetheless, there are presumptions and limitations in the algorithm. The first is that the algorithm is model-based as the AL algorithm in (Abbeel and Ng, 2004), which requires that the environment can be modeled as an MDP with known transition function. In robotics control tasks, this algorithm requires that the dynamics must be known beforehand. The second is that, as AL algorithm, in every iteration a policy must be solved optimally with respect to a reward function. This requires using reinforcement learning algorithm over the entire state space in every iteration. The third is that the scalability of probabilistic model checking and counterexample generation is still an open question. Another limitation is that this algorithm does not guarantee the global optimum of the final output and the convergence does not depend on optimum of learnt policy but on the multi-objective parameter.

In the next chapter, we will use experiments to evaluate the effectiveness of the algorithm.

# Chapter 4

# Experiments

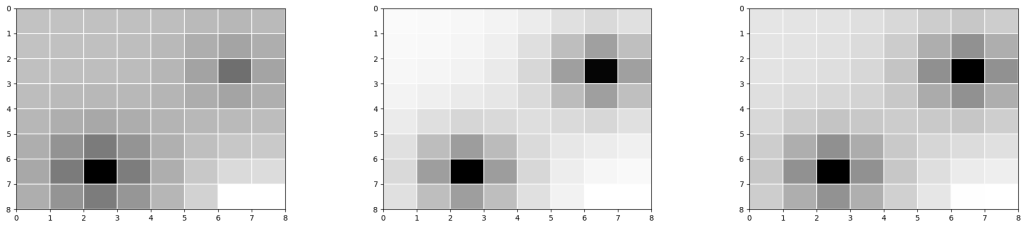
We evaluate our algorithm on three case studies: (1) grid-world, (2) cart-pole, and (3) mountain-car. The cart-pole environment[1] and the mountain-car environment[2] are obtained from OpenAI Gym. All experiments are carried out on a quad-core i7-7700K processor running at 3.6 GHz with 16 GB of memory. Our prototype tool was implemented in Python[3]. For the OpenAI-gym experiments, in each step, the agent sends an action to the OpenAI environment and the environment returns an observation and reward. The parameters are $\gamma = 0.99, \sigma = 10^{-5}, \alpha = 0.5$. For grid-world, $\epsilon = 1E - 5$, whereas for OpenAI-gym environments, $\epsilon = 10$. We use model checking results to show that our algorithm can guarantee the safety of the learnt policy. Moreover, we quantitatively show that the learnt policy has comparable performance compared with the one directly learnt using AL.

## 4.1 Navigation in Grid World

Our first experiment is an extension on the 2D navigation task in Section 3.1. Specifications with different $p^*$'s in are given. Fig. 4·1 shows the different reward mappings that induce the policies learnt by our algorithm. As the safety threshold (value of $p^*$) decreases, the algorithm will try to find a weight vector that assigns low rewards to the unsafe states and states around them, so that the agent will have a lower probability

[1] https://github.com/openai/gym/wiki/CartPole-v0
[2] https://github.com/openai/gym/wiki/MountainCar-v0
[3] https://github.com/zwc662/CAV2018

of moving into the unsafe areas. However, as a result, the agent will also focus more on avoiding the unsafe areas than actually reaching the goal area. In essence, we trade off performance with safety. We also evaluated the scalability of our implementation



(a) (b) (c)

**Figure 4·1:** (a)$p^* = 90\%$. The learnt policy has a probability of 12.5% with respect to the given PCTL property. The reward function assigns lower rewards to the unsafe states than the states nearby. (b) $p^* = 40\%$. The learnt policy has a probability of 11.0% with respect to the given PCTL property. The reward function assigns even lower rewards to the unsafe states, indicated by the greater contrast between unsafe states and other states. (c)$p^* = 10\%$. The learnt policy has a probability of 4% with respect to the given PCTL property. The reward function assigns such low rewards to the unsafe states that the states nearby are also assigned with low rewards (because of the radial basis feature functions).

**Table 4.1:** Average runtime per iteration in seconds.

| Size | Num. of States | Compute $\pi$ | Compute $\mu$ | MC | Cex |
|---|---|---|---|---|---|
| $8 \times 8$ | 64 | 0.02 | 0.02 | 1.39 | 0.014 |
| $16 \times 16$ | 256 | 0.05 | 0.05 | 1.43 | 0.014 |
| $32 \times 32$ | 1024 | 0.07 | 0.08 | 3.12 | 0.035 |
| $64 \times 64$ | 4096 | 6.52 | 25.88 | 22.877 | 1.59 |

using the grid-world example. The first and second columns indicate the size of the grid world and the resulting state space. The third column shows the average runtime that policy iteration takes to compute an optimal policy $\pi$ for a known reward function. The forth column indicates the average runtime that policy iteration takes to compute the expected features $\mu$ for a known policy. The fifth column indicates the average runtime of verifying the PCTL formula using PRISM. The last column

indicates the average runtime that generating a counterexample using COMICS.

## 4.2 Cart-Pole from OpenAI Gym

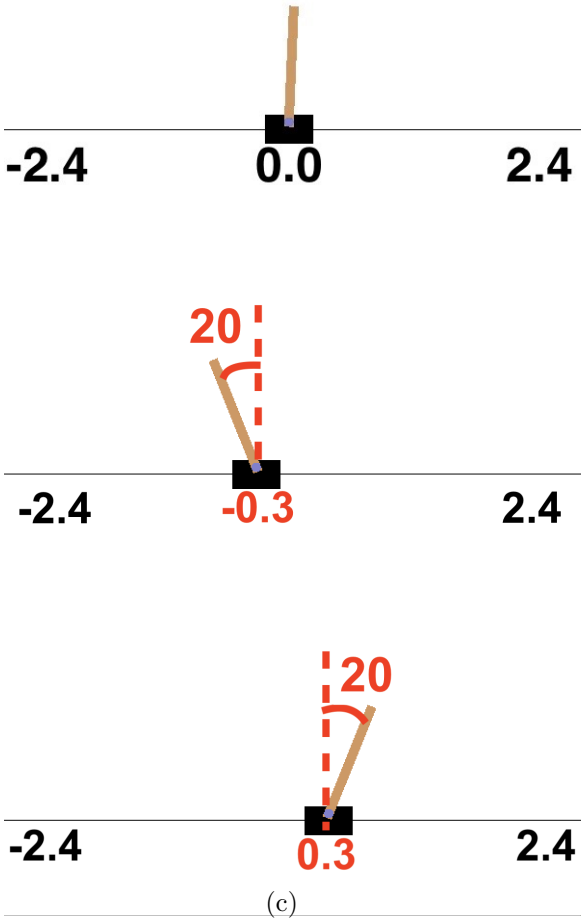



**Figure 4·2:** (a) The cartpole environment. (b) The cart is at -0.3 and pole angle is -20°. (c) The cart is at 0.3 and pole angle is 20°.

In grid world, it is hard to evaluate the performance that the agent may need to sacrifice for safety. Hence we implement the algorithm in OpenAI gym environments where performance can be quantified. In the cart-pole environment as shown in Fig. 4·2(a), the goal is to keep the pole on a cart from falling over as long as possible by moving the cart either to the left or to the right in each time step. The maximum

step length is $t = 200$. The position, velocity and angle of the cart and the pole are continuous values and observable, but the actual dynamics of the system are unknown.

We discretize the continuous observation space and formulate the environment as an MDP with 400 states and 2 actions. Through exploring the environment, the transition function is determined by the samples of experienced transitions. The feature vector in each state contains 30 radial basis functions which depend on the squared Euclidean distances between current states and other 30 states which are uniformly distributed in the state space. In addition, a maneuver is deemed *unsafe* if the pole angle is larger than 20° while the cart's horizontal position is more than ±0.3 as shown in Fig. 4·2(b) and 4·2(c). We formalize the safety requirement in PCTL as (4.1).

$$\Phi ::= P_{\leq p^*}[\mathit{true}\ \mathbf{U}^{\leq t}\ (\mathit{angle} \leq -20^\circ \wedge \mathit{position} \leq -0.3) \vee (\mathit{angle} \geq 20^\circ \wedge \mathit{position} \geq 0.3)] \tag{4.1}$$

**Table 4.2:** In the cart-pole environment, *higher* average steps mean better performance. The safest policy is synthesized using PRISM-games.

| | MC Result | Avg. Steps | Unsafe Rate | Num. of Iters |
|---|---|---|---|---|
| AL | 49.1% | 165 | 19% | 2 |
| Safest Policy | 0.0% | 8 | 0.0% | N.A. |
| $p^* = 30\%$ | 17.2% | 121 | 13.0% | 9 |
| $p^* = 25\%$ | 9.3% | 136 | 17.0% | 13 |
| $p^* = 20\%$ | 17.2% | 122 | 10.8% | 8 |
| $p^* = 15\%$ | 7.3% | 138 | 15.4% | 21 |
| $p^* = 10\%$ | 7.2% | 136 | 13.7% | 21 |
| $p^* = 5\%$ | 0.04% | 83 | 0.5% | 50 |

We consider only demonstrations for which the pole is held upright without violating any of the safety conditions for all 200 steps. The safest policy synthesized by PRISM-games is used as the initial safe policy. We also compare the different policies learned by CEGAL for different safety threshold $p^*$s. In Table 4.2, the policies are

compared in terms of model checking results ('MC Result') on the PCTL property in (4.1) using the constructed MDP, the average steps ('Avg. Steps') that a policy (executed in the OpenAI environment) can hold across 5000 rounds (the higher the better), and averaged percentage times ('Unsafe Rate') that a policy (executed in the OpenAI environment) violates the $unsafe$ conditions across 5000 rounds. The last column in the table shows the averaged number of iterations for these algorithms to converge (with 50 as the maximum number of iterations). The policy in the first row is the result of using AL alone. Observe that from $p^* = 30\%$ to 10%, the performance of the learnt policy is similar. However, when the safety threshold becomes very low, e.g., $p^* = 5\%$, the performance of the learnt policy drops significantly. Observe also that the safest policy has the lowest performance amongst all. It corresponds to simply letting the pole fall and thus does not risk moving the cart out of the range [-0.3, 0.3]. We note that the discrepancy between the 'MC result' and the 'unsafe rate' is due to the MDP abstraction of the actual game environment. The safety guarantee of the algorithm is based on the former and the latter is used for validation purposes. The discordances between model checking results and unsafe rates are due to the inaccurate transition function. Despite the inconsistency, the policies learnt via CEGAL still show lower frequency of reaching unsafe states than that via AL.

## 4.3 Mountain-Car from OpenAI Gym

Our third experiment uses the mountain-car environment from OpenAI Gym. As shown in Fig. 4·3(a), a car starts from the bottom of the valley and tries to reach the mountaintop on the right as quickly as possible. In each time step the car can perform one of the three actions, accelerating to the left, coasting, and accelerating to the right. The agent fails if the step length reaches the maximum ($t = 66$). The velocity and position of the car are continuous values and observable while the exact dynamics are unknown.

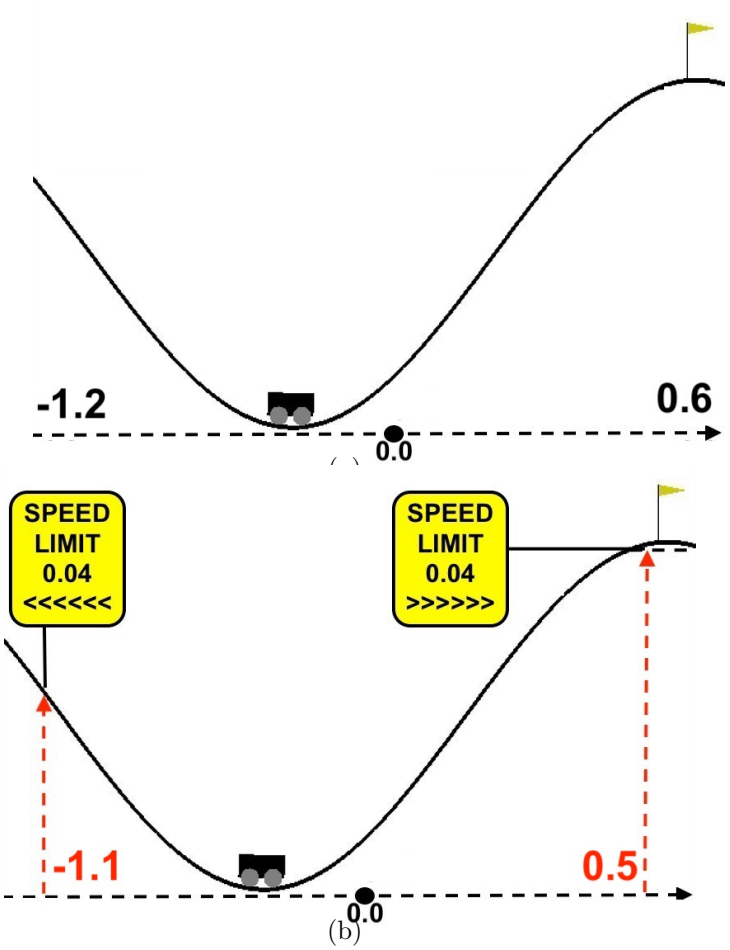



**Figure 4·3:** (a) The original mountain-car environment. (b) The mountain-car environment with traffic rules: when the distance from the car to the left edge or the right edge is shorter than 0.1, the speed of the car should be lower than 0.04.

We discretize the continuous observation space and formulate the environment as an MDP with 320 states and 3 actions. Through exploring the environment, the transition function is determined by the samples of experienced transitions. The feature vector for each state contains 2 exponential functions and 18 radial basis functions which respectively depend on the squared Euclidean distances between the current state and other 18 states which are uniformly distributed in the state space. In this game setting, the car cannot reach the right mountaintop by simply accelerating to the right. It has to accumulate momentum first by moving back and forth in the valley. The safety rules we enforce are shown in Fig. 4·3(b). They correspond to

speed limits when the car is close to the left mountaintop or to the right mountaintop (in case it is a cliff on the other side of the mountaintop). Similar to the previous experiments, we consider only expert demonstrations that successfully reach the right mountaintop without violating any of the safety conditions. The average step length of demonstrations is 40. We formalize the safety requirement in PCTL as (4.2).

$$\Phi ::= P_{\leq p^*}[true\ \mathbf{U}^{\leq t}\ (speed \leq -0.04 \wedge position \leq -1.1) \\ \vee\ (speed \geq 0.04 \wedge position \geq 0.5)] \tag{4.2}$$

**Table 4.3:** In the mountain-car environment, *lower* average steps mean better performance. The safest policy is synthesized via PRISM-games.

| | MC Result | Avg. Steps | Unsafe Rate | Num. of Iters |
|---|---|---|---|---|
| Policy Learnt via AL | 69.2% | 54 | 100% | 50 |
| Safest Policy | 0.0% | *Fail* | 0% | 0 |
| $p^* = 60\%$ | 43.4% | 57 | 33.2% | 9 |
| $p^* = 50\%$ | 46.9% | 55 | 29.4% | 23 |
| $p^* = 40\%$ | 29.3% | 61 | 0.6% | 25 |
| $p^* = 30\%$ | 18.9% | 64 | 0.0% | 17 |
| $p^* = 20\%$ | 12.0% | 66 | 3.5% | 39 |
| $p^* = 10\%$ | 7.6% | *Fail* | 0% | 40 |

We compare the different policies using the same set of categories as in the cart-pole example. The numbers are averaged over 5000 rounds. The last column in the table shows the averaged number of iterations for these algorithms to converge (with 50 as the maximum number of iterations). As shown in the first row, the policy learnt via AL has the highest probability of going over the speed limits. We observe that this policy makes the car speed up all the way to the left mountaintop to maximize its potential energy. The safest policy corresponds to simply staying in the bottom of the valley. The policies learnt via CEGAL for the safety threshold $p^*$ ranges from 60% to 50% not only have lower probability of violating speed limits but also maintain comparable performance. However, when the safety threshold $p^*$ further decreases,

the agent becomes more conservative and it takes more time for the car to finish the task. The discordances between model checking results and unsafe rates are due to the inaccurate transition function. Despite the inconsistency, the policies learnt via CEGAL still show obvious restraint in reaching unsafe states than that via AL.

## 4.4 Discussion

In the three experiments, we evaluate our algorithm in different aspects. From the gridworld experiment, we observe how safety specification influences the implicit search for reward function in our algorithm. When the safety threshold decreases, lower rewards will be assigned to the unsafe states so that the optimal policy with respect to the reward function will avoid reaching those states. From cart-pole and mountain-car experiments, we observe how our algorithm guarantee the safety of the final output policy while retaining the performance of the learnt policy in the mean time. In both experiments, by learning from human demonstrations and the counterexamples for violating the safety specification, the agent not only knows how to finish the tasks but also exhibits the awareness of safety.

# Chapter 5

# Conclusions

## 5.1 Summary of the thesis

In this thesis, a counterexample-guided approach is proposed for combining formal verification with apprenticeship learning to ensure safety of the learning outcome. By giving a safety specification and adding a verification oracle to the original apprenticeship learning algorithm, it is guaranteed that only a policy that satisfies the safety specification will be output. Furthermore, when a learnt policy is verified to violate the safety specification, a counterexample, which is a proof of the violation, will be extracted from the policy. Without having to risk deploying the unsafe policy in the field, a counterexample can be regarded as a set of negative examples. The approach in this thesis makes novel use of the counterexamples to steer the policy search process by formulating the problem as a multi-objective optimization problem. By using an adaptive weight approach, the multi-objective optimization problem is solved iteratively. The multi-objective weight parameter is updated according to the learnt policy in each iteration. The algorithm is guaranteed to terminate when the multi-objective weight parameter converges to a certain value. The experimental results indicate that the proposed approach can guarantee safety and retain performance for a set of benchmarks including examples drawn from OpenAI Gym.

In addition to what have been achieved in this thesis, there are some open challenges in the theories of the algorithm. Firstly, this thesis does not guarantee finding

a safe policy that performs as well as expert policy. Although it is common sense that being safe can sometimes conflicts with having high performance, finer control in leverage between different objects would be beneficial. More specifically, when solving the multi-objective optimization problem, although an adaptive weight sum approach is employed, the algorithm does not end up with the Pareto frontier. Although the linear constraints ensures the dominance of the optimal $\pi$, $\hat{\pi}$ and $cex$ in individual iteration, more candidate policies and counterexamples will be found as the iteration continues. Thus, the dominance in one iteration may not hold in the future iteration. One shortcoming of the thesis is in the termination of the algorithm which is fully based on the multi-objective parameter rather than converging to the global optimum of the solved policy. There are also challenges in model construction which brings the gap between the margin of policy features and margin of the true policy performance. In OpenAI gym environments, where performance can be quantified, a learnt policy that is $\epsilon$-close to the expert policy does not necessarily have performance quantitatively comparable with expert policy. Whereas, this requires dedicated feature design for the reward function as well as more accurate transition function, which is out of the scope of this thesis.

## 5.2 Future Work

In the future, firstly we plan to deploy our framework in tasks more complex than the OpenAI gym environments, such as ROS[1] environment and Baxter[2] robot.

Secondly, we will test the scalability of the current verification techniques utilized in our prototype. Different methods in policy verification and counterexample generation will also be considered in case of necessity of improving the efficiency of the framework. For instance, we are considering applying statistical model check-

[1] http://wiki.ros.org/ROS/Tutorials/InstallingandConfiguringROSEnvironment
[2] http://www.rethinkrobotics.com/baxter/

ing (Younes et al., 2011), (Younes and Simmons, 2006), (Henriques et al., 2012) in large scale learning tasks where the state space may be too large for symbolic model checking techniques.

In addition, we will concentrate on addressing the theoretical shortcomings in the CEGAL algorithm. For instance, we will consider using gradient based method as in (Neu and Szepesvári, 2012), rather than iteration based method, to solve apprenticeship learning problem or other imitation learning problems, such as the model-free method in (Ho et al., 2016). Meanwhile, we will investigate new ways to incorporate counterexample in the learning process.

# References


Abbeel, P. and Ng, A. Y. (2004). Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, pages 1–, New York, NY, USA. ACM.

Ábrahám, E., Jansen, N., Wimmer, R., Katoen, J.-P., and Becker, B. (2010). Dtmc model checking by scc reduction.

Alshiekh, M., Bloem, R., Ehlers, R., Könighofer, B., Niekum, S., and Topcu, U. (2017). Safe reinforcement learning via shielding. *CoRR*, abs/1708.08611.

Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., and Mané, D. (2016). Concrete problems in AI safety. *CoRR*, abs/1606.06565.

Bellman, R. (1957). A markovian decision process. 6:15.

Bundy, A. (2017). Preparing for the future of artificial intelligence. *AI Soc.*, 32(2):285–287.

Fujita, M., McGeer, P. C., and Yang, J.-Y. (1997). Multi-terminal binary decision diagrams: An efficient data structure for matrix representation. *Formal methods in system design*, 10(2-3):149–169.

Gillulay, J. H. and Tomlin, C. J. (2011). Guaranteed safe online learning of a bounded system. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 2979–2984. IEEE.

Han, T., Katoen, J. P., and Berteun, D. (2009). Counterexample generation in probabilistic model checking. *IEEE Transactions on Software Engineering*, 35(2):241–257.

Hansson, H. and Jonsson, B. (1994). A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6(5):512–535.

Held, D., McCarthy, Z., Zhang, M., Shentu, F., and Abbeel, P. (2017). Probabilistically safe policy transfer. *CoRR*, abs/1705.05394.

Henriques, D., Martins, J. G., Zuliani, P., Platzer, A., and Clarke, E. M. (2012). Statistical model checking for markov decision processes. In *Quantitative Evaluation of Systems (QEST), 2012 Ninth International Conference on*, pages 84–93. IEEE.

Ho, J., Gupta, J., and Ermon, S. (2016). Model-free imitation learning with policy optimization. In *International Conference on Machine Learning*, pages 2760–2769.

Jansen, N., Ábrahám, E., Scheffler, M., Volk, M., Vorpahl, A., Wimmer, R., Katoen, J., and Becker, B. (2012). The COMICS tool - computing minimal counterexamples for discrete-time markov chains. *CoRR*, abs/1206.0603.

Jha, S. and Seshia, S. A. (2017). A theory of formal synthesis via inductive learning. *Acta Informatica*, 54(7):693–726.

Junges, S., Jansen, N., Dehnert, C., Topcu, U., and Katoen, J.-P. (2016). Safety-constrained reinforcement learning for mdps. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 130–146. Springer.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Kuo, Y.-J. and Mittelmann, H. D. (2004). Interior point methods for second-order cone programming and or applications. *Computational Optimization and Applications*, 28(3):255–285.

Kwiatkowska, M., Norman, G., and Parker, D. (2002). Prism: Probabilistic symbolic model checker. *Computer Performance Evaluation/TOOLS*, 2324:200–204.

Kwiatkowska, M. and Parker, D. (2013). *Automated Verification and Strategy Synthesis for Probabilistic Systems*, pages 5–22. Springer International Publishing, Cham.

Kwiatkowska, M., Parker, D., and Wiltsche, C. (2017). Prism-games: verification and strategy synthesis for stochastic multi-player games with multiple objectives. *International Journal on Software Tools for Technology Transfer*.

Levinson, J., Askeland, J., Becker, J., Dolson, J., Held, D., Kammel, S., Kolter, J. Z., Langer, D., Pink, O., Pratt, V., et al. (2011). Towards fully autonomous driving: Systems and algorithms. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pages 163–168. IEEE.

Mason, G. R., Calinescu, R. C., Kudenko, D., and Banks, A. (2017). Assured reinforcement learning for safety-critical applications. In *Doctoral Consortium at the 10th International Conference on Agents and Artificial Intelligence*. SciTePress.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Moldovan, T. M. and Abbeel, P. (2012). Safe exploration in markov decision processes. *arXiv preprint arXiv:1205.4810*.

Neu, G. and Szepesvári, C. (2012). Apprenticeship learning using inverse reinforcement learning and gradient methods. *arXiv preprint arXiv:1206.5264*.

Ng, A. Y. and Russell, S. J. (2000). Algorithms for inverse reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pages 663–670, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Puggelli, A., Li, W., Sangiovanni-Vincentelli, A. L., and Seshia, S. A. (2013). Polynomial-time verification of pctl properties of mdps with convex uncertainties. In *Proceedings of the 25th International Conference on Computer Aided Verification*, CAV'13, pages 527–542, Berlin, Heidelberg. Springer-Verlag.

Ratliff, N. D., Bagnell, J. A., and Zinkevich, M. A. (2006). Maximum margin planning. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 729–736, New York, NY, USA. ACM.

Sadigh, D., Kim, E. S., Coogan, S., Sastry, S. S., and Seshia, S. A. (2014). A learning based approach to control synthesis of markov decision processes for linear temporal logic specifications. In *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*, pages 1091–1096. IEEE.

Shiarlis, K., Messias, J., and Whiteson, S. (2016). Inverse reinforcement learning from failure. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, pages 1060–1068. International Foundation for Autonomous Agents and Multiagent Systems.

Solar-Lezama, A., Tancau, L., Bodik, R., Seshia, S., and Saraswat, V. (2006). Combinatorial sketching for finite programs. *SIGOPS Oper. Syst. Rev.*, 40(5):404–415.

Wimmer, R., Jansen, N., Ábrahám, E., Becker, B., and Katoen, J.-P. (2012). *Minimal Critical Subsystems for Discrete-Time Markov Models*, pages 299–314. Springer Berlin Heidelberg, Berlin, Heidelberg.

Younes, H. L. S., Clarke, E. M., and Zuliani, P. (2011). Statistical verification of probabilistic properties with unbounded until. In Davies, J., Silva, L., and Simao, A., editors, *Formal Methods: Foundations and Applications*, pages 144–160, Berlin, Heidelberg. Springer Berlin Heidelberg.

Younes, H. L. S. and Simmons, R. G. (2006). Statistical probabilistic model checking with a focus on time-bounded properties. *Inf. Comput.*, 204(9):1368–1409.

Ziebart, B. D., Maas, A., Bagnell, J. A., and Dey, A. K. (2008). Maximum entropy inverse reinforcement learning. In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3*, AAAI'08, pages 1433–1438. AAAI Press.

# Weichao Zhou

(857) 250-7548 Email: zwc662@gmail.com
3 Ashford Ct, Allston, Boston, MA 02134, USA
LinkedIn: https://www.linkedin.com/in/zhou-weichao-83a16bb6/
Github: https://github.com/zwc662

## RESEARCH INTERESTS

- Artificial Intelligence, Reinforcement Learning
- Formal Methods

## EDUCATION

**Boston University**

- M.S in Computer Engineering 09/2016-present

**Fudan University**

- M.S in Microelectronics 09/2012-06/2015

**Fudan University**

- B.S in Microelectronics 09/2008-06/2012

## WORK & RESEARCH EXPERIENCE

**Safety-Aware Inverse Reinforcement Learning** 09/2016-present

- Conducting researches on combining formal verification and inverse reinforcement learning to build up a safety-aware AI framework.

**Micron Technology, Inc.**, Product Engineer 05/2015-07/2016

- Designed test cases and wrote test scripts in Python to verify eMMC and UFS products.
- Wrote test programs and scripts in C++ and Perl to test NAND memory circuits.

**TFET's (Tunneling Field-Effect Transistor) Application on Memory** 03/2012-01/2014

- Cooperated with semiconductor chip manufacturer to fabricate TFET units and DRAM cells based on TFETs.

**FinFET (Fin Field-Effect Transistor) Project** 01/2014-06/2015

- Used Sentaurus to simulate a new FinFET structure and used SPICE to simulate circuits based on the new FinFETs.

**High frequency AlGaN/GaN HEMT (High-electron-mobility Transistor) on Silicon Substrate** 05/2014-06/2015

- Fabricated high-frequency AlGaN/GaN HEMTs on Si substrate for RF circuit applications.

**Evaluation of Buffer Organizations for Network-on-Chip** 09/2010-05/2011

- Applied queueing theory to analyze Network-on-Chips performance under different allocations of virtual channels and channel buffer resources.

## PUBLICATIONS

1. **Weichao Zhou,** Wenchao Li, "**Safety-Aware Apprenticeship Learning**", International Conference on Computer-Aided Verification (CAV), 2018
2. **Weichao Zhou**, Peng-Fei Wang, Zhang, D.W., **"A sub-10nm U-shape FinFET design with suppressed leakage current and DIBL effect,"** in Semiconductor Technology International Conference (CSTIC), 2015 China , vol., no., pp.1-3, 15-16 March 2015 doi: 10.1109/CSTIC.2015.7153322
3. **Weichao Zhou**, Xi Lin, Xiao-Yong Liu, Xiang-Ming Xu, Chun-Min Zhang, Jin-Shan Shi, Peng-Fei Wang, Zhang, D.W., "**Investigation of spin-on-dopant for fabricating high on-current tunneling field effect transistor**," in

Solid-State and Integrated Circuit Technology (ICSICT), 2014 12th IEEE International Conference on , vol., no., pp.1-3, 28-31 Oct. 2014 doi: 10.1109/ICSICT.2014.7021392

4. Ming'e Jing, Pengshuai Ren, **Weichao Zhou**, Zhiyi Yu, Xiaoyang Zeng, "**Evaluation of buffer organizations for network-on-chip**," in Solid-State and Integrated Circuit Technology (ICSICT), 2012 IEEE 11th International Conference on , vol., no., pp.1-3, Oct. 29 2012-Nov. 1 2012 doi: 10.1109/ICSICT.2012.6467882