IMPERIAL COLLEGE LONDON, COMPUTING DEPARTMENT

MEng Individual Project Interim Report: Safety-Aware Multi-Agent Apprenticeship Learning

Junchen Zhao

January 23, 2020

CONTENTS

1 Introduction

Person in Supervision of this project: Belardinelli Francesco, Borja Gonzalez

1.1 Project Background

As the rapid development of Artifical Intelligence in the current technology field, the Reinforcement Learning has been proven to be a powerful technique that allows autonomous agents to learn optimal behaviors (called policies) in unknown and complex environments through models of rewards and penalizations. However, in order to make this technique (Reinforcement Learning) work correctly and get the precise reward function which returns feedback to the learning agent about when the agent behaves correctly or not, the reward function needs to be thoroughly specified.

As a result, in real-world complex environments, such as autonomous driving, specifying a correct reward function could be one of the hard tasks to tackle for the Reinforcement Learning model designers. To this end, Apprenticeship Learning techniques such as Inverse Reinforcement Learning, in which technique the agent infers a reward function from expert behaviors, are of high interest due to the fact that they could result in highly specified reward function efficiently.

However, for critical tasks such as autonomous driving, we need to critically consider about the safety-related issues, so as to we need to build techniques to automatically check and ensure that the inferred rewards functions and policies that resulted from the Reinforcement Learning model system fulfill the needed safety requirements of the critical tasks that we have mentioned previously.

1.2 Project Objective

In order to have a well-designed Reinforcement Learning model system, which is able to generate the highly-specified reward function and ensure the safety-related considerations, we are applying the Reinforcement Learning technique mentioned in the background section called Apprenticeship Learning, which will be introduced in detail in the later section.

Our objective of this project is to make the extra extension based on the technique mentioned in the paper **"Safety-aware Apprenticeship Learning"** written by **Weichao Zhou and Wenchao Li**[3] to improve the utility and the efficiency of the existing Reinforcement Learning model system from single-agent learning setting to multi-agent setting framework.

In the paper **"Safety-aware Apprenticeship Learning"**, the latent techniques include the **Probabilistic Computational Tree Logic** as the way of model checking and the **Apprenticeship Learning**:

1. **Probabilistic Computational Tree Logic .** as the way of model checking:
    - According to the paper **"Safety-Aware Apprenticeship Learning "**, PCTL 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%". As a result, PCTL allows for probabilistic quantification of properties, a technique which is also called probabilistic Model checking and can be applied to the policy quantification checking process in reinforcement learning. [3]

2. **Apprenticeship Learning**:
   - Essentially the Apprenticeship Learning is a kind of learning from demonstration techniques where the reward function of a Markov Decision Process is unknown to the learning agent. At the same time, the agent has to derive a good policy by observing an expert's demonstrations according to the paper "Safety-Aware Apprenticeship Learning" by Weichao Zhou and Wenchao Li.[3]

I will give detailed explanations about the latent techniques about how the safety-aware Apprenticeship learning works, the place that we will add the extension features to the current technique, and where our current progress is.

1.3 My Contribution to the Project

1. Recently, work has been done regarding extracting and correcting Inverse Reinforcement Learning policies in the single-agent domain. Regarding to the fact that we will add extension to the Inverse Reinforcement Learning system from single-agent setting to multi-agent setting, My first contribution to this project is considering the case of extracting safe reward functions from expert behaviors in multi-agent systems instead of being from the single-agent system.
2. My second contribution is extending the single-agent system framework to multi-agent system framework. As a result, I will focus on extending this or similar frameworks to multi-agent Inverse Reinforcement Learning and design a novel framework in the end.
3. My final contribution to this project is evaluating empirically the performance of my extension to the single-agent Inverse Reinforcement Learning framework.

2 Project Prerequisite Knowledge

In this section, I will give detailed background explanation and introduction about the latent and prerequisite concepts about the Apprenticeship learning for the purpose of understanding our project objectives. I will summarize the bullet points that I will cover in the this section in the following parts:

1. (2.1) Definition of Apprenticeship Learning
2. (2.2) Reinforcement Learning Basic(Markov Decision Process)
3. (2.3)Basic properties of Markov Decision Process
4. (2.3) Inverse Reinforcement Learning Basic
5. (2.4) Probabilistic Computational Logic Tree (PCLT) for Model Checking
6. (2.5) Markov Game

2.1 Definition of Apprenticeship Learning

We consider the formulation of **Apprenticeship Learning(AL)** by Abbeel and Ng[1]:

1. The concept of AL is closely related to **reinforcement learning (RL)** where an agent learns what actions to take in an environment (known as a policy) by maximizing some notion of long-term reward.
2. In AL, however, the agent is not given the reward function, but instead has to first estimate it from a set of expert demonstrations via a technique called **inverse reinforcement learning**.
3. The formulation assumes that the reward function is expressible as a linear combination of known state features.
4. An expert demonstrates the task by maximizing this reward function and the agent tries to derive a policy that can match the feature expectations of the expert's demonstrations. Apprenticeship learning can also be viewed as an instance of the class of techniques known as **Learning from Demonstration (LfD)**.

As a result, essentially the Apprenticeship Learning is a kind of learning from demonstration techniques where the reward function of a Markov Decision Process is unknown to the learning agent. At the same time, the agent has to derive a good policy by observing an expert's demonstrations according to the paper "Safety-Aware Apprenticeship Learning" by Weichao Zhou and Wenchao Li.[3]

- ***Unknown Reward Function***: We consider the setting where the unknown reward function to the agent in the Markov Decision Process is assumed to be a linear combination of a set of state features.

It's possible for someone who can get confused about the definition of the Apprenticeship Learning if he or she lacks of the background in Reinforcement Learning and Inverse Reinforcement Learning. Therefore, I will give detailed explanation about the Reinforcement Learning basic and the Inverse Reinforcement Learning basic in the follow subsections.

2.2 Reinforcement Basics(Markov Decision Process)

In this section, I will talk about the reinforcement basics and mainly focus on the Markov Decision Process.

Based on the definition of the Reinforcement Learning from Wikipedia, the broad definition of **Reinforcement Learning** can be defined as such below:

- ***Reinforcement Learning***: Reinforcement learning is an area of machine learning concerned with how software agents ought to take actions in an environment in order to maximize some notion of cumulative reward. Reinforcement learning is one of three basic machine learning paradigms, alongside supervised learning and unsupervised learning.

In Reinforcement Learning, one of the founding concepts is the **Markov Decision Process(MDP)**. In order to understand it, I will introduce the MDP briefly in the following paragraph.

Markov Decision Process broadly can be defined as a discrete time stochastic control process. It provides a mathematical framework for modeling decision making in situations where outcomes are partly random and partly under the control of a decision maker.

However, specifically the MDP can be defined as a finite tupe, which contains five components $\{S, A, P, \gamma, s_0, R\}$, composed a process following a set of actions, which are named as **Policies** π. I will make the explanation about the meanings of these five terminologies and what the **Policy** π is below:

1. **S** is a finite set of states;
2. **A** is a set of actions;
3. **P** is a transitional probability function describing the probability of transitioning from one state **s**, which belongs to the state set **S**, to another state by taking action **a**, which belongs to the action set **A**;
4. **R** is the reward function which maps each state **s**, which belongs to the state set **S**, to a real number indicating the reward of being in state **s**;
5. s_0 is the initial state of the MDP which belongs to the state set **S** as well;
6. γ is the discount factor which describes how future rewards attenuate when a sequence of transitions is made;
7. π is defined as any mapping from **S** to **A**, so as to the **Value Function** for a policy π evaluated at any state **s** can be defined as:

$$V^{\pi}(s) = E[R(s_1) + \gamma \cdot R(s_2) + \gamma^2 \cdot R(s_3) + ... + |\gamma] \tag{2.1}$$

In the formula shown above, where the expectation is over the distribution of the state sequence (s_1,s_2, s_3...) we pass through when we execute the policy π starting from s_1. We also define the **Q-function**, according to :

$$Q^{\pi}(s,a) = R(s) + \gamma \cdot E_{s'\ P}[V^{\pi}(s')] \tag{2.2}$$

The goal of standard reinforcement learning is to find a policy π such that V^{π} , **the value function in the Markov Decision Process**, could be maximized.

2.3 Basic Properties of Markov Decision Process

After I talked about how the Markov Decision Process works in the previous detailed setting, I will give a brief introduction to the **basic properties of MDPs**. The basic properties of MDPs could generally be summarized by 2 theorems.[2]

Theorem One (Bellman Equation): Let an MDP {S, A, P, γ, s_0, R} and a policy π: S->A be given. Then, for all s ∈ S, a∈ A, V^{π} and Q^{π} satisfy

$$V^{\pi}(s) = R(s) + \gamma \cdot \sum_{s'} P_{s\pi(s)}(s') \cdot V^{\pi}(s') \tag{2.3}$$

$$Q^{\pi}(s,a) = R(s) + \gamma \cdot \sum_{s'} P_{sa}(s') \cdot V^{\pi}(s') \tag{2.4}$$

Theorem Two (Bellman Optimality): Let an MDP {S, A, P, γ, s_0, R} and a policy π: S->A be given. Then π is an optimal policy for M iff, for all s∈ S

$$\pi(s) \in arg\max_{a \in A} Q^{\pi}(s,a) \tag{2.5}$$

2.4 Inverse Reinforcement Learning Basics

After the introduction about the standard reinforcement learning and the Markov Decision Process, we have generally understood how the reinforcement learning works. Now, I'm going to give introduction and detailed explanation about the **Inverse Reinforcement Learning(IRL)**. [2]

According to Andrew Ng, the IRL problem is to find a reward function that can explain observed behavior. By applying the IRL technique, we aim at recovering the reward function R in the MDP tuple which we mentioned in the previous subsection MDP {S, A, P, γ, s_0, R} from a set of m trajectories demonstrated by an expert.

Based on the setting mentioned above in which we'll recover the reward function R in the MDP tuple from a set of m trajectories demonstrated by an expert, we have the IRL from sampled m Monte Carlo trajectories.

We assume that we have the ability to simulate m trajectories (m_0 , m_1 , m_2 , ...) in the Markov Decision Process from the initial state s_0 under the optimal policy π^* or any policy of our choice. For each policy π that we will consider, including the optimal policy π^*, we will need a way of estimating the $V^{\pi}(s_0)$ for any setting of the α s, where α s is the unknown parameter that we want to "fit" in the linear function approximation.

- In order to achieve this goal of estimating the $V^{\pi}(s_0)$, we first execute the m sampled Monte Carlo trajectories under π.
- Then, for each $i = 1, \dots, d$, if R = r_i, define $V_i^{\pi}(s_0)$ to be the average empirical return that would have been on these m Monte Carlo trajectories.
- For example, if we only take $m = 1$ trajectories, and the trajectory visited the sequence of states $(s_0, s1, \dots)$, then we have the formula below:

$$\hat{V_i}^{\pi}(s_0) = r_i(s_0) + \gamma \cdot r_i(s_1) + \gamma^2 \cdot r_i(s_2) + \dots \tag{2.6}$$

As what we have seen above, if we take m number of sampled trajectories, then the $V^{\pi}(s_0)$ will be the average over the empirical returns of m such trajectories. Then for any setting of the α_i s, a natural estimate of $V_i^{\pi}(s_0)$ is:

$$\hat{V_i}^{\pi}(s_0) = \alpha_1 \cdot \hat{V_i}^{\pi}(s_0) + \alpha_2 \cdot \hat{V_i}^{\pi}(s_0) + \dots\dots\dots + \alpha_d \cdot \hat{V_i}^{\pi}(s_0) \tag{2.7}$$

By describing the detail about how to recover the reward function R in the MDP tuple from a set of m trajectories by using IRL. We can finally explain the corresponding algorithm in detail.

- First, we find the value estimates as described above for the assumed optimal policy π^* that we are given and the random policy that we randomly choose π_1.
- The inductive step is as follow:
  1. We have a set of policies $\{\pi_1, \dots., \pi_k\}$
  2. We want to find a setting of the α_i s so that the resulting reward function can satisfy as follow:

$$\hat{V_i}^{\pi^*}(s_0) \geq V^{\pi_i}(s_0), i, \dots., k \tag{2.8}$$

2.5 Probabilistic Computational Logic Tree (PCLT) for Model Checking

According to the paper **"Safety-Aware Apprenticeship Learning "**, PCTL 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%". As a result, PCTL allows for probabilistic quantification of properties, a technique which is also called probabilistic Model checking and can be applied to the policy quantification checking process in reinforcement learning. [3]

In order to figure out how the PCTL works for the model checking, we need to figure out what the syntax of PCTL is. In PCTL, there are two main syntaxes, including the **State Formulas** and the **Path Formulas**.

First, let's understand what the **State Formulas** syntax is.

1. Generally, we use symbol ϕ to represent the **State Formulas**.
2. **State Formulas** asserts the property of a single state $s \in S$ in the MDP.
3. ϕ ::= true $|l_i| \neg\, \phi_i\, |\phi_i \wedge \phi_j| P_{\rhd\lhd p*}(\psi)$.
4. $\rhd \lhd \in \{\leq, \geq, >, <\}$.

5. $P_{\rhd\lhd p*}(\psi)$ means that the probability of generating a trajectory that satisfies the formulas ψ, which is the **Path formulas** and we will talk about it in below, is $\rhd \lhd p^*$.

Second, let's understand what the **Path formulas** syntax is.

1. Generally, we use symbol ψ to represent the **Path formulas**.
2. **Path Formulas** asserts the property of a trajectory.
3. $\psi ::= X\ \phi\ |\phi_1\ U^{\leq k}\ \phi_2\ |\phi_1\ U\ \phi_2$.
4. $X\ \phi$ asserts that the next state after initial state in the trajectory satisfies ϕ.
5. $\phi_1\ U^{\leq k}\ \phi_2$ asserts that ϕ_2 is satisfied in at most k transitions and all preceding states satisfy ϕ_1.
6. $\phi_1\ U\ \phi_2$ asserts that ϕ_2 will be eventually satisfied and all preceding states satisfy ϕ_1.
7. The semantics of PCTL is defined by a satisfaction relation $\models$ as follows.
   - s $\models$ true iff state $s \in S$.
   - s $\models \phi\ iff$ state s satisfies the **State formula** ϕ.
   - $\tau \models \psi\ iff$ trajectory τ satisfies the **Path formula** ψ .

After briefly talking about what the syntax of PCTL is, we can make more detailed explanation about how the PCTL is implemented to the Apprenticeship Learning process as a model checker on the policy.

2.6 Markov Game

Due to the reason that we are going to improve the current learning system from single agent to multi-agents, Markov Game is needed in this process. In order to understand how it works, I will give detail explanation about it in this subsection.

Basically, **Markov Games** are the generalization of the Markov Decision Processes(MDPs) to the case of N interacting agents and a **Markov Game** is defined as (S, A, P, η, r) via:

1. a set of states S.
2. N sets of actions $(A_i)_{i->N}$.
3. The function P: $S \times A_1 \times A_2 \times ... \times A_N \longrightarrow P(S)$ describes the stochastic transition process between states, where $P(S)$ means the set of probability distributions over the set S.
4. By giving that we are in state s^t at time t, and the agent takes actions $\{a_1$,......, $a_N\}$, the state transitions to s^{t+1} with probability $P(s^{t+1}|s^t, a_1, ..., a_N)$.
5. By taking the actions, each agent i obtains a bounded reward given by a function r_i: $S \times A_1 \times A_2 \times ... \times A_N \longrightarrow R$.
6. The function $\eta \in P(S)$ specifies the distribution of the initial state.

Now, by giving the basic definition about the **Markov Game**, then we can use the bold variables without subscription i to denote **the concatenation of all variables for all agents**:

- For example, **a** denotes actions of all agents and **r** denotes all rewards in multi-agent setting.

Then, we use the subscript $-i$ to denote all agents except for the agent i:

- For example, (a_i, a_{-i}) represents $(a_1, ..., a_N)$, which is action of all number of N agents.

The objective of each agent i in the multi-agent setting is to maximize the its expected return:

1. The expected return of the agent is defined as: $E_\pi\ [\sum_{t=1}^{N} \gamma^t r_{i,t}]$:
   - γ is the discount factor.
   - $r_{i,t}$ is the reward received t steps into the future.
2. Each agent in the Markov Game can achieve its own objective by selecting actions through a stochastic policy $\pi_i : S \longrightarrow P(A_i)$.
   - Then depending on the context, the policies can be Markovian or require additional coordination signals.
3. Finally, based on all of the terms that I explained before in this subsection, we can, for each agent i, finally further define the expected return for a state-action pair as:
   - $ExpRet_i^{\pi_i,\pi_{-i}}\ (s_t, a_t) = E_{s^{t+1:T}, a^{t+1:T}}\ [\sum_{l \geq t} \gamma^{l-t} r_i(s^l, a^l) | s_t, a_t, \pi]$
     a) π_i, π_{-i}: policies of all number of N agents.
     b) T: total number of steps.
     c) l: The Future step.
     d) t: The current step.
     e) s_t: The state at current step t.
     f) a_t: The agent action at current step t.
     g) i: This denotes the current agent.

3 Safety-Aware Apprenticeship Learning Explanation

Person in Supervision of this project: Belardinelli Francesco, Borja Gonzalez

By giving the detailed introduction about prerequisite knowledge for understanding our project in the previous section. In this chapter, I will put the main focus on how the **Single agent Safety-Aware Apprenticeship Learning** works due to the reason that our objective main objective is extending the single agent learning system framework to multi-agent learning system framework. I will summarize the bullet points that I will cover in the this section in the following parts:

1. (3.1) Recap of Apprenticeship Learning.
2. (3.2) Apprenticeship Learning via Inverse Reinforcement Learning.
3. (3.3) PCTL Model Checking in Apprenticeship Learning.
4. (3.4) The Framework for Safety-aware Apprenticeship Learning.
5. (3.5) Counterexample-Guided Apprenticeship Learning Algorithm.
6. (3.6) Problem Solved by Safety-Aware Apprenticeship Learning.

3.1 Recap of Apprenticeship Learning

Again, I will restate what the apprenticeship learning is, how it normally works, and why we should apply it in this subsection for the purpose of recapping the content I've mentioned in the previous section.

As I've mentioned before, Apprenticeship learning is a kind of learning from demonstration techniques where the reward function of a Markov decision process is unknown to the learning agent. At the same time, the agent has to derive a good policy by observing an expert's demonstrations. In the **Apprenticeship Learning** setting, we consider the setting where the **unknown reward function** to the agent in the MDP is assumed to be a **linear combination of a set of state features**, which would be discussed in detail in the following subsection.

Regarding to whom is reading this report, you must be curious about why it's called safety-aware apprenticeship learning. The reason is due to the fact that the rapid progress of artficial intelligence (AI) comes with a growing concern over its safety when deployed in real-life systems and situations. If the objective function of an AI agent is wrongly specified, then maximizing that objective function may lead to harmful results. In addition, the objective function or the training data may focus only on accomplishing a specific task and ignore other aspects, such as safety constraints, of the environment. As a result, adding a safety specification framework expressed in Probabilistic Computational Logic Tree, which was mentioned in previous section, as a way of model checking to the learning algorithm, which is the Apprenticeship Learning algorithm, is important. By embedding the probabilistic model checking inside Apprenticeship Learning, we can ensure safety while retaining the performance of the learnt policy.

In the following subsections, I'm going to talk about how the Apprenticeship learning algorithm is achieved via which kind of technique.

3.2 Apprenticeship Learning via Inverse Reinforcement Learning

Because we have introduced in section two about the Inverse Reinforcement Learning(**IRL**), which is finding a reward function that can explain observed behavior and essentially recovering the reward function R in the MDP, we can explain how the IRL is applied to the Apprenticeship learning in this subsection.

Inverse Reinforcement Learning(IRL) aims at recovering the reward function R of MDP {S, A, P, γ, s_0, R} from a set of m trajectories $\tau = \{\tau_1 , \tau_2 , \tau_3, ...\}$ demonstrated by the experts. In order to achieve AL via IRL, AL assumes that the reward function R of MDP is linear combination of state features. such as $R(s) = w^T\ f(s)$.[3]

1. $f : S \implies [0,1]^k$ is a vector of known features over states S.
2. $w \in R^k$ is an unknown weight vector that satisfies $||w_2|| \leq 1$.

By knowing the AL assumptions showing above, we can estimate the **expected features** of a policy π, the **expected features** which are **expected values** of the **cumulative discounted state features** $f(s)$ by following π on M, such that $\mu_E = E\ [\sum_{t=0}^{\infty} \gamma^t\ f(s_t) \mid \pi\ ]$.

1. μ_E denotes the expected features of the **unknown expert's policy** π_E.
2. μ_E can be approximated by the expected features of expert's m demonstrated trajectories: $\mu_E = 1/m \sum_{\tau \in \tau_E} \sum_{t=0}^{\infty} \gamma^t\ f(s_t)$, if the set of demonstrated trajectories by the expert in the size of m is big enough.

As a result, given a error bound ϵ, **a policy** π^* is defined to be ϵ-close to the **unknown expert's policy** π_E:

- If the **expected feature** $\mu\pi^*$ satisfies the relation that: $||\mu_E - \mu_\pi^*||_2 \leq \epsilon$.
- The expected features of the policy $\mu\pi^*$ can be calculated by the **Monte Carlo Method**, **value iteration** or **Linear Programming**.

In order to calculate the expected features of a policy $\pi^*$and find the optimal policy π, we are going to use the algorithm proposed by Abbeel and Ng starts with a **random policy** π_0 and its **expected policy** μ_{π_0}.

1. Assuming in iteration i, we have found a set i candidate policies $\Pi = \{\pi_0, \pi_1, \pi_2, ...\}$ and the corresponding expected features $\{\mu_\pi \mid \pi \in \Pi\}$, then:

$$\delta = \max_{w} \min_{\pi \in \Pi} w^T(\hat{\mu_E} - \mu_\pi) s.t. ||w||_2 \leq 1 \tag{3.1}$$

- The **optimal** w is used to find the corresponding **optimal policy** π_i and the **expected features** μ_{π_i}.
- If $\delta < \epsilon$, which is the error bound between the **expected feature** from the ***unknown expert's policy*** π_E and the ***current policy*** π_i:
  1. The algorithm terminates, and the policy π_i is produced as the optimal policy.
  2. Otherwise, the **expected feature** from the current expert's policy, which is $\mu_{\pi_i}$is

added to the set of features for the policy set Π and the algorithm continues the iteration until the optimal policy is found.

3.3 PCTL Model Checking in Apprenticeship Learning

In the previous section, I have introduced how the PCTL works as a way of model checking, so I will assume you have familiarized with the terminologies and the concepts for understanding the following contents. Now, I'm going to explain how the PCTL Model Checking works in Apprenticeship Learning.

Based on the Zhou and Li's algorithm for model checking[3], they define the $pref(\tau)$ as the set of all $prefixes$ of trajectory τ including τ itself, then $\tau \models_{min} \psi$ ($\models_{min}$ means there is minimal satisfaction relationship exists) ***iff***

$$(\tau \models \psi) \wedge (\forall \tau^{'} \in pref(\tau) \setminus \tau, \tau^{'} \not\models \psi) \tag{3.2}$$

- In a easier way to explain this satisfaction relationship, we can utilize an example as such:
  1. if $\psi = \phi_1\ U^{\leq k}\ \phi_2$, then for any finite trajectory, we have the minimal satisfaction relationship exists that:
     - $\tau \models_{min} \phi_1\ U^{\leq k}\ \phi_2$, and only the final state in τ satisfies ϕ_2.

Therefore, let $P(\tau)$ be the **Probability of transitioning along the trajectory** τ and let τ_{ψ} be the set of all finite trajectories that satisfies $\tau \models_{min} \psi$ (This relationship is explained above), then the value of **PCTL** property ψ is defined as $\sum_{\tau \in \tau_{\psi}} P(\tau)$.

- So, for a **Discrete-time-Markov-Chain (DTMC)** M_{π} and a **state formula**, we have:
  1. A *counterexample* of ϕ is a set $cex \subseteq \tau_E$ that satisfies $\sum_{\tau \in cex} P(\tau) > p^*$.
  2. $P(\tau) = \sum_{\tau^* \in \tau} P(\tau^*)$ is the sum of probability of all trajectories in trajectory set τ.
  3. $CEX_{\phi} \subseteq 2^{\tau_{\psi}}$ is the set of all *counterexamples* for a formula ϕ.

3.4 The Framework for Safety-aware Apprenticeship Learning

Based on Zhou and Li[3], the framework for safety-aware Apprenticeship Learning can be concluded to the process shown in the figure **(3.1)**. Based on this figure, we can generalize the framework as such simplified version in text below:

1. We utilize Information from both of $Verifier$ and $ExpertDemonstration$.
   a) **Performing the Model Checking**: $Verifier$ check the candidate policy π^* satisfies the **State Formula** ϕ or not.
   b) If candidate policy π^* satisfies the **State Formula** ϕ, then:
      i. Check whether our learning objective is met. Our learning objective is to check whether the $\delta < \epsilon$, where δ is the optimal difference between the expected feature from expert demonstration policy and the expected feature

from the current policy, and ϵ is the error bound that we defined previously.

A. If we meet our learning objective, then the **optimal policy** is generated.

B. Otherwise, we add the current candidate policy to the policy set Π.

c) If candidate policy π^* doesn't satisfy the **State Formula** ϕ, then:

i. We generate the counterexample cex.

ii. Continuing the iteration.

d) The iteration will continue unless the optimal policy is found.

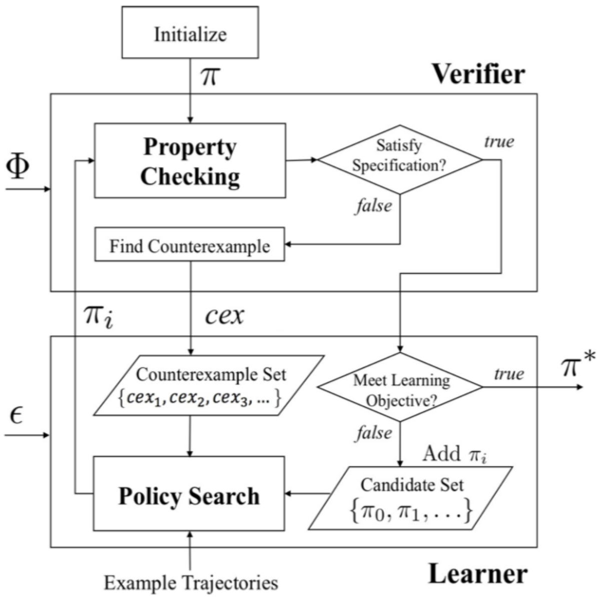


Figure 3.1

After discussing the basic framework for the safety-aware Apprenticeship Learning, we can dive into more details about how this framework works.

By looking at the claims from Zhou and Li[3], the basic AL algorithm can be finding a weight w under the condition that the expected reward π_E maximally outperforms any mixture of the policies in the candidate policy set Π.

As a result, we can view the weight w as the normal vector of hyperplane $w^T\ (\mu - \mu_E) = 0$, which has the maximal distance to the convex hall of the set $\{\mu_E \mid \pi \in \Pi\}$. So, we can show that:

1. $w^T\ \mu_\pi \geq w^T\ \mu_{\pi^i}$ for all found policy π.
2. Intuitively, by performing the this kind of **max-margin separation technique**, we can move the candidate policy's expected feature closer to the expected feature demonstrated by expert.
3. We can explain this abstract concept in the image below.

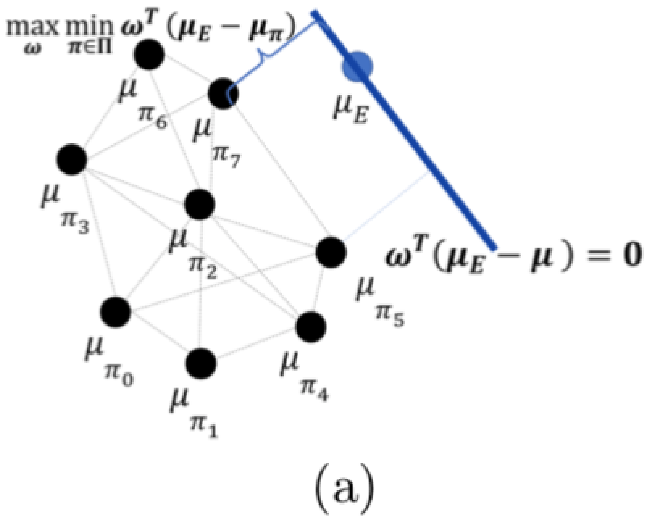
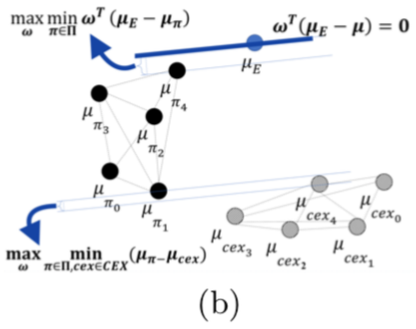


Figure 3.2

Therefore, we can similarly apply this technique, the max-margin separation principle technique, to maximize the distance between the **candidate policy** and the **Counterexamples**.

1. Let $CEX = \{cex_0, cex_1, cex_2, ...\}$ denotes the set of counterexamples of the policies that do not satisfy the Specification Φ in the framework.
2. Maximizing the distance between the convex hulls of the set $\{\mu_{cex} \mid cex \in CEX\}$ and the set $\{\mu_\pi \mid \pi \in \Pi\}$ is equivalent to maximizing the distance between the parallel supporting hyperplanes of the two convex hulls.
3. The corresponding image to illustrate the abstract concept is shown below:



$\max_{\omega} \min_{\pi\in\Pi} \omega^T(\mu_E - \mu_\pi)$
$\omega^T(\mu_E - \mu) = 0$
μ_E
μ_{π_4}
μ_{π_3}
μ_{π_2}
μ_{π_0}
μ_{π_1}
μ_{cex_4}
μ_{cex_0}
μ_{cex_3}
μ_{cex_2}
μ_{cex_1}
$\max_{\omega} \min_{\pi\in\Pi, cex\in CEX} (\mu_\pi - \mu_{cex})$
(b)

Figure 3.3

- This is the corresponding formula:

$$\delta = \max_{w} \min_{\pi\in\Pi, cex\in CEX} w^T(\mu_\pi - \mu_{cex}) s.t. ||w||_2 \leq 1 \tag{3.3}$$

Finally, in order to attain good performance similar to that of the expert, we still want to learn from μ_E. Thus, the overall problem can be formulated as a multi-objective optimization problem, and formulate to the formular show below:

$$\max_{w} \min_{\pi\in\Pi, cex\in CEX, \hat{\pi}\in\Pi} (w^T(\mu_\pi - \mu_{cex}), w^T(\mu_{\hat{\pi}} - \mu_{cex})) s.t. ||w||_2 \leq 1 \tag{3.4}$$

3.5 Counterexample-Guided Apprenticeship Learning Algorithm

Finally, we come to the final stage of the safety-aware Apprenticeship Learning, which is the counterexample-guided apprenticeship learning algorithm used for solve SafeAL problem. We can regard this algorithm as a special case of the framework of the safety-aware apprenticeship learning shown above.

The one special add-on in this case to the original framework is using **adaptive weighting scheme** to weight the scheme from μ_E with the separation from μ_{cex}.

Originally, our framework works like this:

$$\max_{w} \min_{\pi\in\Pi, cex\in CEX, \hat{\pi}\in\Pi} (w^T(\mu_\pi - \mu_{cex}), w^T(\mu_{\hat{\pi}} - \mu_{cex})) s.t. ||w||_2 \leq 1 \tag{3.5}$$

Now, our framework works like this:

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

Figure 3.4

- The K and $(1-K)$ are our **weighting scheme** as an add-on to the original framework.
- Assuming $\Pi_S = \{\pi_1, \pi_2, ...\}$ is a set of candidate policies that satisfy the **Specification** Φ.
- Assuming $CEX = \{cex_1, cex_2, ...\}$ is a set of counterexamples.
- We introduce a parameter K into the **formula (3.5)** and change it into a weighted sum of optimization problem shown in the **figure (3.4)**.
- It's important to note: π and $\hat{\pi}$ are different:
  1. The optimal weight vector w solved from figure (3.4) can be use for generating the new policy π_w by using algorithm such as policy iteration.
  2. Then, we apply the probabilistic model checker to see **if** π_w **satisfies** Φ:
     a) Satisfy: We add the newly generate π_w to the candidate policy set Π_s.
     b) Not satisfy: We generate a counterexample cex_{π_w} and add it to the counterexample set CEX.

Detailed algorithm pseudo-code is shown above:

```
Algorithm 1 Counterexample-Guided Apprenticeship Learning (CEGAL)
 1: Input:
 2:     $M \leftarrow$ A partially known $MDP\backslash R$; $f \leftarrow$ A vector of feature functions
 3:     $\mu_E \leftarrow$ The expected features of expert trajectories $\{\tau_0, \tau_1, \ldots, \tau_m\}$
 4:     $\Phi \leftarrow$ Specification; $\epsilon \leftarrow$ Error bound for the expected features;
 5:     $\sigma, \alpha \in (0,1) \leftarrow$ Error bound $\sigma$ and step length $\alpha$ for the parameter $k$;
 6: Initialization:
 7:     If $||\mu_E - \mu_{\pi_0}||_2 \leq \epsilon$, then return $\pi_0$          ▷ $\pi_0$ is the initial safe policy
 8:     $\Pi_S \leftarrow \{\pi_0\}$, $CEX \leftarrow \emptyset$          ▷ Initialize candidate and counterexample set
 9:     $inf \leftarrow 0, sup \leftarrow 1, k \leftarrow sup$          ▷ Initialize multi-optimization parameter $k$
10:     $\pi_1 \leftarrow$ Policy learnt from $\mu_E$ via apprenticeship learning
11: Iteration $i$ ($i \geq 1$):
12:     Verifier:
13:         status $\leftarrow Model\_Checker(M, \pi_i, \Phi)$
14:         If status = SAT, then go to Learner
15:         If status = UNSAT
16:             $cex_{\pi_i} \leftarrow Counterexample\_Generator(M, \pi_i, \Phi)$
17:             Add $cex_{\pi_i}$ to $CEX$ and solve $\mu_{cex_{\pi_i}}$, go to Learner
18:     Learner:
19:         If status = SAT
20:             If $||\mu_E - \mu_{\pi_i}||_2 \leq \epsilon$, then return $\pi^* \leftarrow \pi_i$
21:                                                   ▷ Terminate. $\pi_i$ is $\epsilon$-close to $\pi_E$
22:             Add $\pi_i$ to $\Pi_S$, $inf \leftarrow k, k \leftarrow sup$          ▷ Update $\Pi_S$, $inf$ and reset $k$
23:         If status = UNSAT
24:             If $|k - inf| \leq \sigma$, then return $\pi^* \leftarrow \underset{\pi \in \Pi_S}{argmin}||\mu_E - \mu_\pi||_2$
25:                                         ▷ Terminate. $k$ is too close to its lower bound.
26:             $k \leftarrow \alpha \cdot inf + (1-\alpha)k$          ▷ Decrease k to learn for safety
27:         $\omega_{i+1} \leftarrow \underset{\omega}{argmax} \underset{\pi \in \Pi_S, \tilde{\pi} \in \Pi_S, cex \in CEX}{\min} \omega^T(k(\mu_E - \mu_\pi) + (1-k)(\mu_{\tilde{\pi}} - \mu_{cex}))$
28:     ▷ Note that the multi-objective optimization function recovers AL when $k = 1$
29:         $\pi_{i+1}, \mu_{\pi_{i+1}} \leftarrow$ Compute the optimal policy $\pi_{i+1}$ and its expected features
    $\mu_{\pi_{i+1}}$ for the MDP $M$ with reward $R(s) = \omega_{i+1}^T f(s)$
30:     Go to next iteration
```

Figure 3.5

- In this algorithm, $sup = 1$, which is a constant.
- inf, is a variable, and $inf \in [0, sup]$ for the upper and lower bound respectively.
- The learner determines the value of k within the bound $[inf, sup]$ in each iteration depending on the outcome of the $Verifier$ and use k to solve the line 27 in the **figure (3.5)**.

Based on this algorithm, we can produce a general theorem showing below[3]:

1. Given the initial policy π_0 that satisfies the **Specification** Φ, this **Counterexample-Guided Apprenticeship Learning Algoritm** promises:
    a) Producing a policy π^*: such π^* in **formula (3.1)** satisfies Φ.
    b) and such π^* has the performance is at least as good as that of π_0 when compare with π_E.

3.6 Problem Solved by Safety-Aware Apprenticeship Learning

1. **Definition of safety issue in apprenticeship learning:** An agent following the learnt policy would have a higher probability of entering those unsafe states than it should.
2. **Reasons of having the safety issue in apprenticeship learning:**
    a) Expert policy π_E itself has a high probability of reaching the unsafe states.
    b) Human expert often tend to perform only successful demonstrations that do not highlight the unwanted situations. (Lack of negative examples)
3. **The safety-aware apprenticeship learning problem :**
    a) Given an MDP, a set of m trajectories demonstrated by an expert, and a **specification (Φ)**, to learning a policy that satisfies the state formula and is $error-closed$ to the expert policy π_E.

4 Summary

This interim report includes the overall summarization about what the project background is and what our future plan on this project is. In the following months, I will keep updating this report until this project is successfully completed in the end of May.

Thanks a lot to all of the support from my co-supervisor Professor elardinelli Francesco and Borja Gonzalez. If there weren't their support, it would be impossible for me to come so far.

References

[1] P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, 2004.

[2] A. Y. Ng and S. Russell. Algorithms for Inverse Reinforcement Learning. *in Proc. 17th International Conf. on Machine Learning*, pages 663–670, 2000.

[3] W. L. Weichao Zhou. Safety-Aware Apprenticeship Learning. *Computer Aided Verification (CAV) 2018*, 1, 2018.