Performance Metrics for State-Based Imitation Learning

Mohamed Zalat, Babak Esfandiari
Carleton University
1125 Colonel By Drive, Ottawa, ON
Canada, K1S 5B6

Abstract

We propose five new domain-independent metrics for evaluating and comparing performance at imitating a state-based expert. We use two agents in the RoboCup environment to compare the performance metrics: an agent based on a Multi-Layer Perceptron (MLP) and an agent based on a Long Short-Term Memory (LSTM) neural network.

1 Introduction

Motivation

In imitation learning, it is not uncommon to use domain specific metrics for evaluating the performance of an agent at imitating an expert (Ross, Gordon, and Bagnell 2011; Abbeel, Coates, and Ng 2010). When this metric is simply a score of the imitation learning agent in its application domain as in (Ross, Gordon, and Bagnell 2011), we are making the assumption that the expert's goal is to maximize or minimize that specific metric in its domain. However, such metrics are evaluating how well the agent is performing in the domain, as opposed to how well the agent is imitating the expert. An example of a domain specific metric that would fail to capture how well an agent is imitating an expert is the number of goals an agent scores in a soccer simulation. In this example, an agent may not be imitating the expert properly and score more goals than the expert, making the metric inadequate in capturing how well the agent imitates the expert.

In other cases where the domain of the expert contains no such metric for evaluating the performance of the agent, it is common to resort to metrics such as the F1-score or the accuracy of the agent with respect to the expert (Gunaratne, Esfandiari, and Fawaz 2018; Tîrnăucă et al. 2016; Ontanón, Montaña, and Gonzalez 2013; Floyd and Esfandiari 2011). One main issue with those metrics, despite them adequately measuring how well an agent imitates a reactive expert (an expert that only relies on the most recent environment state to take a decision), is that they do not take into account that the prediction problem is a sequential problem when the expert is state-based (i.e. relies on previous environment states and actions in addition to the most recent environment state). Moreover, if a test set of pre-collected trajectories of the expert is used for testing, those metrics do not give information on how well the agent imitates the expert in practice, as the agent will visit a much different environment state distribution once it takes an action the expert would not take. This is because slightly deviating from the expert's behavior may result in the learner visiting states the expert rarely/never visits and further deviating from the expert's behavior as a result. To address this problem, we propose new metrics to compare the agent's trajectory to the expert's trajectory in the test set instead of measuring the accuracy of the agent.

A metric for measuring the deviation between two trajectories was proposed in (Tîrnăucă et al. 2016). However, this distance metric was used to compute the distance between a learning model's input and output pair with the corresponding pairs in the expert's trajectory. Their use of this distance metric was model-centric as opposed to being centered on the trajectories. Therefore, in models that take the current environment state as their input, this metric would only measure the distance between the reactive component of the agent's and the expert's behavior. Hence, its value will not reflect whether the agent is taking actions based on the same reasons the expert took those actions (which may be from past environment states). This points to the lack of state-based imitation learning metrics that are model and domain independent. Ideally, those metrics should be based on the following 2 components: the trajectory before the agent or expert took the action, and the action the agent or expert took as a result.

Contributions

The domain we use to compare our proposed metrics to existing metrics is the RoboCup 2D virtual soccer simulation environment (Itsuki 1995). The RoboCup environment provides each agent with angle and distance information of each object in its field of view. We explain the RoboCup environment in more detail in Section 5. We propose five metrics that use the chi-squared statistic to measure the deviation between different distributions of the agent and the expert

derived from their trajectory. Traditionally, it is used to test the independence of a pair of random variables based on observations of the pair. This makes it suitable for measuring the distance between two distributions.

To test the validity of the proposed metrics, we use two experts: a reactive expert and a state-based expert; and two agents: a Multi-Layer Perceptron (MLP) agent that is only capable of learning reactive behavior and a Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber 1997) agent that is capable of learning state-based behavior.

We also compare our proposed metrics with commonly used metrics and assess the behavior of the agents qualitatively. We argue that some of our proposed metrics are better at comparing the performance of imitation learning agents when they are imitating state-based behavior than common metrics. In addition, our metrics have a stronger statistical ground that enables us to use them in a chi-squared test to reject or fail to reject the hypothesis stating that the agent's and expert's distributions are the same.

Structure

The formal definition of an agent, its trajectory, and imitation learning are presented in Section 2. In Section 3, we briefly discuss the state of the art metrics that measure how well an agent learns state-based behavior. Section 4 introduces our proposed metrics based on the chi-squared statistic between the agent and the expert. We describe the RoboCup environment and action space, the behavior of our two experts, the architecture of our learning agents and the size of the data set used for each agent in Section 5. In Section 6, we discuss our results and compare our performance metrics in the cases of imitating reactive behavior and state-based behavior.

2 Background

Imitation Learning

Imitation learning is the process of learning an expert's behavior using example trajectories of the expert, and possibly the expert's feedback during the execution of the learning agent (Argall et al. 2009). The environment of an agent can be defined by a discrete set of environment states S; where an agent selects an action a from the set of actions A the agent can perform (Wooldridge 2009).

Trajectories

A trajectory of an agent's run is simply a sequence of environment state and action pairs. For the rest of this paper we will use the words run and trajectory interchangeably. (Wooldridge 2009) represents a run using the following notation:

$$r : s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} ... \xrightarrow{a_n} s_n$$

Where the run r starts at the perception of environment state $s_0 \in S$, followed by its action $a_1 \in A$, followed by the perception of s_1, etc. (Wooldridge 2009) defines $\mathcal{R}$ as the set of all finite trajectories. Similarly, we define $\mathcal{R}^S$ as the subset of finite trajectories that end with an environment state and $\mathcal{R}^A$ as the subset of finite trajectories that end with an action.

Agents

An agent can be described as a function that selects an action based on its trajectory ending with the last observed state. (Wooldridge 2009) describes an agent as the following function:

$$Ag : \mathcal{R}^S \rightarrow A$$

We can also simplify the definition of an agent that requires past environment states by representing the information from the past using an internal state $i \in I$. Agents that depend on information from the past are referred to as state-based agents. The following function simplifies the formal definition of an agent:

$$Ag : S \times I \rightarrow A \times I$$

Where I is the finite set of internal states the agent may occupy. This internal state captures the information from past environment states and actions that a state-based agent requires to select an action. However, there are cases where the agent does not need any past information and selects an action purely based on the current environmental state. In those cases, an agent is said to be a reactive agent since it is reacting to the latest observation of its environment. (Wooldridge 2009) defines a reactive agent as the following function:

$$Ag : S \rightarrow A$$

3 State of the Art

It is common to use domain specific metrics, F1-score and accuracy as mentioned previously to estimate how well the agent is imitating the expert (Ross, Gordon, and Bagnell 2011; Abbeel, Coates, and Ng 2010; Gunaratne, Esfandiari, and Fawaz 2018; Tîrnăucă et al. 2016; Ontanón, Montaña, and Gonzalez 2013; Floyd and Esfandiari 2011). However, using domain specific metrics may not necessarily indicate the agent is properly imitating the behavior of the expert. For example, using the number of times a car falls off from a racing course does not necessarily prove that the agent is completely imitating the behavior of the expert driver. On the other hand, F1-score and accuracy are good metrics for indicating how well the agent imitates the expert correctly given we use the expert as an oracle in a test run; however, in cases where a testing set consists of pre-collected expert trajectories, those metrics will show better results than reality. This is because the testing set tests the learner on the expert's environment state distribution as opposed to the learner's environment state distribution. Therefore, it is important to compute the F1-score and accuracy of the agent on its own trajectory, with the expert providing the action it would take in the situations the agent visits. In cases where this is not possible, F1-score and accuracy will not indicate how well the agent imitates the expert when it is deployed in the environment.

Vapnik's risk is a metric that measures the inverse likelihood of a trajectory of an agent to be produced by a particular Probabilistic Finite Automata (PFA). It was used in (Tîrnăucă et al. 2016) with the purpose of behavioral

recognition. They also proposed the Monte Carlo distance metric that does not require a PFA and aims to measure the deviation between the agent's and the expert's trajectory. The Monte Carlo distance is the normalized log of the count of state-action pairs that are identical in trajectories T and T':

$$H(T,T') = -\frac{1}{n}\sum_{i=1}^{n}\log[\frac{\sum_{j=1}^{m}\Pi_{\{o'_j\}}(o_i)+1}{m+|S|\times|A|}]$$

Where $o_i = (s_i, a_i)$ is an observation and action pair in the trained agent's trace T, n is the length of the agent's trace, $o'_j = (s_j, a_j)$ is an observation and action pair in the expert's trace T', m is the length of the expert's trace, and $\Pi_{\{o'_j\}}$ is the indicator function of set $\{o'_j\}$. We include this metric in Section 6, to compare it to our metrics. The observation component was dependent on the model they used. Hence, s_j and s_i consists of both the current and previous environment state in a model that takes the current and previous environment state as input. They used this metric based on three models: a model that takes the current environment state as input; a model that takes the current and previous environment state as input; and a model that takes the current environment state, past environment state and the previous action as input. They perform behavioral cloning (a special case of imitation learning where no feedback from the expert is used) using traditional machine learning models in addition to PFA in a 2D Vacuum cleaner domain.

4 Metrics

Currently all the cross-domain performance metrics used in the field of imitation learning involve measuring the distance between the agent's and expert's $P(A|S)$. For example, the accuracy metric measures the percentage of time the agent selects the correct action at each environment state, and the Monte Carlo distance involves counting the number of identical state-action pairs in the agent's that are present in the expert's trajectory (assuming the model only takes the current environment state as input). However, recall from Section 2 that, unless the agent is reactive, an agent maps an entire trajectory ending with a state to an action. Hence, we can see that all the cross-domain metrics we currently use measure the reactive component of how well an agent imitates an expert. In order to test how well an agent imitates an expert that may be state-based, we need to use metrics that measure the performance of an agent over its $P(A|\mathcal{R}^S)$ distribution (the distribution of actions given its trajectory). Unfortunately, $\mathcal{R}^S$ is a very large set making it impossible to measure the agent's imitation performance over its $P(A|\mathcal{R}^S)$ distribution directly. Also, this probability distribution is likely to be very sparse and sensitive to small variations. However, we can measure it indirectly by using selective sub-sequences from a trajectory.

We propose using $P(S)$, $P(S_t|S_{t-1})$, $P(A)$, $P(A_t|A_{t-1})$ and $P(A|S)$ as estimations of the trajectory of the agent and the expert. We can easily extract those distributions from the trajectory of the agent and the expert and compare them to each other. We use the χ^2 statistic to measure the distance between the two distributions because it is not computationally expensive to compute unlike the Monte Carlo distance. In addition, it allows us to test the indistinguishability of the agent's distribution from the expert's distribution at a chosen confidence level. The χ^2 statistic is calculated using the expected frequency m_i of each possible value of the random variable in the distribution, its observed frequency x_i, and the number of possible values the random variable can take k, using Equation (1). We calculate the χ^2 statistic under the null hypothesis that the agent's distribution and the expert's distribution are independent. In the following subsections, we provide the intuition behind the use of each distribution as an imitation learning metric and how each metric is calculated.

$$\chi^2 = \sum_{i=1}^{k}\frac{(x_i - m_i)^2}{m_i} \tag{1}$$

$\chi^2_{\mathbf{P(S)}}$

We compute this metric by recording the frequency of each unique state visited by the learning agent and the expert in its respective trajectory, then calculating the Chi-squared (χ^2) statistic using the frequency table constructed. Hence, this metric measures the difference between the distribution of states visited by the learning agent in its trajectory with that visited by the expert. This metric should be useful in comparing agents that are imitating state-based behavior, since an agent should be visiting the same environment state distribution of the expert if it is properly imitating the expert regardless of the expert's type. However, it may not necessarily prove that the agent is imitating the expert correctly if there are multiple behaviors that can produce identical $P(S)$ distributions.

$\chi^2_{\mathbf{P(A)}}$

This metric compares the distribution of actions taken in the learning agent's trajectory to that of the expert's trajectory. As opposed to $\chi^2_{P(S)}$, this metric does not guarantee that two agents have the same reward function and as such, it does not guarantee that the agent converged to the expert's behavior. This is because if an agent stochastically selects actions based on the action distribution of the expert without regard of its environment state, it will have the same $P(A)$ as the expert. However, it lets us know at a high level whether the agent is at least imitating the distribution of actions of the expert.

$\chi^2_{\mathbf{P(A|S)}}$

This metric compares the distribution of actions taken at each unique environment state of the agent with that of the expert. This metric is analogous to the accuracy and the F1 score, however, it measures deviation and as such, maximizing accuracy is the same as minimizing $\chi^2_{P(A|S)}$. It is also obvious that this metric is analogous to the concept of the

Monte Carlo distance metric with a model that takes the current environment state as input. As the agent approaches perfect imitation of the expert, the $\chi^2_{P(A|S)}$ metric approaches 0. Again, it may not be ideal to measure how well an agent imitates an expert using this metric if the expert is state-based, since it may have an identical $P(A|S)$ to the expert but does not perform them based on the correct conditions that are required from past states.

$\boldsymbol{\chi^2_{\mathbf{P(S_t|S_{t-1})}}}$

This metric compares the distribution of environment states visited conditioned on the previous environment state of the agent's trajectory with the expert's trajectory. For this metric, we record the frequency of each possible and unique state-to-state transition in the table used to compute the χ^2 statistic. The goal of $\chi^2_{P(S_t|S_{t-1})}$ is to measure how well the agent imitates the state transition distribution of the expert. As opposed to P(S), this metric can prove that the behavior of the agent converged to the expert's behavior if the environment's state transitions depend on the action taken. This is because the agent needs to select the same actions as the expert at the correct states to match the expert's $P(S_t|S_{t-1})$. Therefore, it can be thought of as a stricter version of $\chi^2_{P(S)}$, since it requires the agent to not only visit the same distribution of states as the expert but also to maximize the chance of transitioning to the correct next state (which is achieved by selecting the correct action). Hence, we can say that $\chi^2_{P(S_t|S_{t-1})} \geq \chi^2_{P(S)}$.

$\boldsymbol{\chi^2_{\mathbf{P(A_t|A_{t-1})}}}$

This metric compares the distribution of actions conditioned on the previous action of the agent's trajectory with the expert's trajectory. This makes it useful to compare imitation learning agents when the expert's behavior is state-based as opposed to $\chi^2_{P(A)}$, as the previous action of the agent depends on the previous environment state of the agent. $\chi^2_{P(A_t|A_{t-1})}$ requires the agent to select the correct action in the current environment state which requires correctly selecting the previous action that was selected based on the previous environment state. In cases where the expert is state-based, it will be useful to find out if an agent is not selecting the actions using the same reasoning as the expert or if the agent's model is incapable of capturing the expert's reasoning and instead it is selecting the action randomly.

5 Experiments

In this section, we start by describing the RoboCup domain used to test our metrics followed by the experts we use to train the agents. Lastly, we define the neural networks used for our imitation learning agents, the hyper-parameters of each agent, and the data set used to train each agent on each expert.

RoboCup Environment Space

RoboCup is a virtual soccer simulation environment that provides each agent with details about its environment in a string format. The environment consists of the ball in the soccer field, the two goals, and the other agents in the field (enemy team and friendly team). However, the agents do not have full information about the environment as they are limited by their field of view and as such the environment is partially observable. This poses a challenge in creating a vector that models the perception of an agent, as some information may not be available. Moreover, the RoboCup environment is dynamic and non-deterministic, meaning that the environment may change before the agent takes an action and that its next state is not solely determined by its action and the state it took the action upon.

In order to deal with objects such as the ball and players that may not be in the field of vision of the agent, the environment vector contains a binary field stating if an object is visible or invisible and 2 fields containing the direction and distance information for every possible object in the simulation. When objects are no longer in the field of view, their visibility field is set to 0 and the direction and distance information from the previous vector are used. As we are omitting the position of other agents in the field, the environment vector consists of 9 fields that pertain information about the ball, the enemy goal and the agent's own goal. We do not add other players in the environment vector as we assume all our experts do not use information about other players in the field when making decisions.

RoboCup Action Space

The action space of the agent is discretized to contain only the actions that are performed by the expert that the agent is querying to learn. There are 5 actions the agent can take: dash, turn to the right, turn to the left, turn to the ball, and kick to the goal. The action parameters are pre-defined, yet the agent still has to select the correct action, as the main focus of this paper is to evaluate and validate our metrics.

Experts

To evaluate our metrics we use 2 experts to train each neural network defined in the next section: a reactive expert (Krislet) and a state-based expert (State-Based).

Krislet The expert turns till it finds the ball and runs towards the ball once it finds it. Once it reaches the ball, it locates the enemy goal and tries to kick the ball to the enemy's goal. The expert's name Krislet is after the name of its original code author: Krzysztof Langner.

State-Based Same as the Krislet expert, however, each time it kicks the ball it changes its internal state causing it to turn in the opposite direction when the ball is not in vision (initially it turns in the positive direction when the ball is not in vision, refer to Figure 1).

Neural Networks

To evaluate our metrics we used 2 agents based on different neural networks, one is a simple fully connected MLP and the other is a LSTM. Both neural networks were trained for 200 epochs using the dataset the agents collected at the end of Data Aggregation (DAgger): an imitation learning technique that uses feedback data from the expert during the

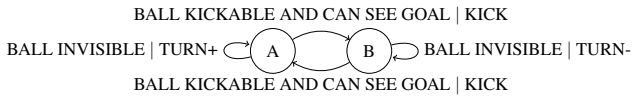

Figure 1: State-Based Expert State Machine: this automaton only contains arrows of environment states that have altered actions when the internal state changes

learning agent's run to maximize its accuracy over its own environment state distribution (Ross, Gordon, and Bagnell 2011). All our neural networks' architectures and training set sizes were selected empirically as the primary goal of our experiments is to test the validity of our metrics. The number of time steps were selected to make the MLP network imitate the reactive expert better than the LSTM network, and to make the LSTM network imitate the behavior of the state-based expert better than the MLP network qualitatively. This is because it was important to qualitatively evaluate how well each neural network imitated each expert in different criteria: to verify that each metric reflects the corresponding difference. The architecture of each neural network and its training procedure are described in the following repository: github.com/MohamedZalat/robocup-soccer.

6 Results

We discretized the environment space of RoboCup to be the set of relevant cases the experts consider when deciding which action to take (excluding information about their internal state) to calculate the Monte Carlo distance and the metrics we are proposing. The cases are mutually exclusive and can be extracted directly from the environment feature vector. For example, one environment state is the agent can see the ball and is in line with the ball, and another environment state is the agent can see the ball but is not in line with the ball. The results of the experiments are presented in Table 1 and Table 2. The goal of our experiments is to test if different sample distributions of a trajectory can be used to learn about the nature of the deviation between the agent's and the expert's behavior, and to validate that our χ^2 metrics are as good of a distance measure as the Monte Carlo distance at comparing the performance of two agents.

Krislet Expert

Looking at the results in Table 1, we can see that the differences within each of our metrics at imitating the Krislet expert are mostly consistent between the MLP and the LSTM agent. The LSTM agent was not able to imitate the behavior of the Krislet expert correctly. Instead, it turns towards the ball once and continuously dashes even if the ball is not perfectly aligned. This is because the expert frequently performs consecutive dashes without requiring to re-align with the ball and the LSTM was able to learn this pattern instead of learning the reason behind performing a dash. Despite this difference in behavior, the LSTM agent consistently defeats the expert. On the other hand, the MLP agent was able to imitate the Krislet expert almost perfectly.

The results of $\chi^2_{P(A)}$, were much lower than all the other metrics for both agents. This is because $P(A)$ is not a difficult distribution to learn from the expert, as the agent could simply be randomly performing actions and would achieve a good $\chi^2_{P(A)}$ score. On the other hand, $\chi^2_{P(S)}$ provides us with meaningful information about the performance of the agents because the agent requires selecting the proper actions at the proper environment states to visit the same environment state distribution (assuming the next environment state depends on the previous action and the previous environment state). Hence, it is larger than $\chi^2_{P(A)}$.

The metrics $\chi^2_{P(A_t|A_{t-1})}$ and $\chi^2_{P(S_t|S_{t-1})}$ can be considered as 'stricter' metrics than $\chi^2_{P(A)}$ and $\chi^2_{P(S)}$. The theoretical reason behind this conclusion is that the agent needs to select the correct action that maximizes the probability of the agent to transition from a state s_i to the next state s_{i+1}; likewise, the agent requires to set itself up for the next correct a_{i+1} with the previous a_i. This was also confirmed by our results $\chi^2_{P(A_t|A_{t-1})} \geq \chi^2_{P(A)}$ and $\chi^2_{P(S_t|S_{t-1})} \geq \chi^2_{P(S)}$. Our results also conclude that it is easier to imitate a reactive expert's $P(A_t|A_{t-1})$ than it is to imitate its $P(S_t|S_{t-1})$.

Lastly, the $\chi^2_{P(A|S)}$ metric was the metric that showed the largest deviation between both agents and the expert. This is due to the sensitivity of the $\chi^2_{P(A|S)}$ metric; any cases where the expert would never do what the agent did greatly influence this metric.

State-Based Expert

The results in Table 2 show the same differences within each of our metrics that we have seen previously in the case of the LSTM agent but this is not the case for the MLP agent. This time the LSTM agent successfully imitated the state-based behavior of the expert and the MLP agent was not able to do so. The MLP agent randomly turns right and left when it can not see the ball instead of consistently turning in one direction. This is how we expected the MLP agent to behave as it is incapable of imitating state-based behavior. Since the LSTM agent results are in line with the same conclusions from the previous section, we will only be discussing the results of the MLP agent.

This time $\chi^2_{P(S)}$ was the lowest metric when using the MLP agent. This is because the MLP agent can still visit a similar environment state distribution to the expert if it is incorrectly selecting the turn direction.

$\chi^2_{P(A_t|A_{t-1})}$ was larger than $\chi^2_{P(S_t|S_{t-1})}$ for the MLP agent, in contrast to its performance with the Krislet expert and the LSTM. When a state-based expert transitions its invisible internal state, it will exhibit the same 'reactive' behavior until it transitions its internal state again. The fact a state-based expert stays in an internal state for a duration of time implies that some $P(a_t|a_{t-1})$ will be much less than others if the agent is properly imitating the expert (i.e. properly transitioning its invisible internal states and properly staying in those internal states). This results in some $P(a_t|a_{t-1})$ that can only occur at internal state transitions, and hence, they are much less likely. In this case, the MLP agent would turn right then left, and the expert never performs this. We may be able to infer that an expert is state-

Agent	Macro F1	Weighted F1	Monte Carlo Distance	$\chi^2_{P(S)}$	$\chi^2_{P(A)}$	$\chi^2_{P(S_t\|S_{t-1})}$	$\chi^2_{P(A_t\|A_{t-1})}$	$\chi^2_{P(A\|S)}$
MLP	0.92	0.96	0.987	14.0	9.24	39.2	33.0	94.7
LSTM	0.5	0.31	6.07	2370	156	3410	478	3700

Table 1: Performance Metrics of Imitating the Krislet Expert

Agent	Macro F1	Weighted F1	Monte Carlo Distance	$\chi^2_{P(S)}$	$\chi^2_{P(A)}$	$\chi^2_{P(S_t\|S_{t-1})}$	$\chi^2_{P(A_t\|A_{t-1})}$	$\chi^2_{P(A\|S)}$
MLP	0.58	0.78	2.50	276	336	366	739	816
LSTM	0.77	0.93	1.39	152	103	205	150	289

Table 2: Performance Metrics of Imitating the State-Based Expert

based by comparing $\chi^2_{P(A_t|A_{t-1})}$ with $\chi^2_{P(S_t|S_{t-1})}$ in agents that are incapable of learning state-based behavior but this requires more experimentation.

When using the weighted F1-score or the Monte-Carlo distance of the MLP agent to evaluate its ability to replicate the behavior of the State-Based expert, one may be convinced that the agent succeeded at learning the behavior of the expert. However, if one were to use the $\chi^2_{P(A_t|A_{t-1})}$ score, they would be able to see that the variance between the $P(A_t|A_{t-1})$ distribution of the expert's and the agent's trajectories is too large for the agent to be capable of learning the behavior of the State-Based expert. More concretely, it highlights the fact that the MLP agent is not properly switching its turning direction based on its internal state. Instead, the agent is turning in random directions.

7 Conclusion

In this paper, we introduced five new metrics that measure the deviation between an imitation learning agent's trajectory and an expert's trajectory, by using distributions extracted from the trajectories instead of relying on the expert as an oracle during testing. We were able to show how $\chi^2_{P(S_t|S_{t-1})}$ and $\chi^2_{P(A_t|A_{t-1})}$ metrics can be used to provide us with more qualitative information about how well an agent is able to imitate state-based behavior compared to traditional metrics. Our results also highlighted the importance of measuring the deviation using the distribution that is characteristic to the state-based expert being imitated, such as the $P(A_t|A_{t-1})$ in the context of the State-Based expert (as it will never turn right and then turn left). This provides us with a more reliable metric to evaluate or track whether an agent is learning the behavior of an expert compared to the F1-score.

An area where future work can be done is evaluating our metrics on a state-based expert that has more internal states, to see how the metrics we proposed compare to standard metrics. Another area is how well our metrics fare in a χ^2 test to test the indistinguishability of a selected distribution between two trajectories.

References

Abbeel, P.; Coates, A.; and Ng, A. Y. 2010. Autonomous helicopter aerobatics through apprenticeship learning. *The International Journal of Robotics Research* 29(13):1608–1639.

Argall, B. D.; Chernova, S.; Veloso, M.; and Browning, B. 2009. A survey of robot learning from demonstration. *Robotics and autonomous systems* 57(5):469–483.

Floyd, M. W., and Esfandiari, B. 2011. Learning state-based behaviour using temporally related cases. In *Nineteenth UK Workshop on Case-Based Reasoning, Cambridge, UK*, volume 9.

Gunaratne, A. S. E.; Esfandiari, B.; and Fawaz, A. 2018. A case-based reasoning approach to learning state-based behavior. In Brawner, K., and Rus, V., eds., *Proceedings of the Thirty-First International Florida Artificial Intelligence Research Society Conference, FLAIRS 2018, Melbourne, Florida, USA. May 21-23 2018*, 377–382. AAAI Press.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Itsuki, N. 1995. Soccer server: a simulator for robocup. In *JSAI AI-Symposium 95: Special Session on RoboCup*. Citeseer.

Ontanón, S.; Montaña, J. L.; and Gonzalez, A. J. 2013. A dynamic bayesian network framework for learning from observation. In *Conference of the Spanish Association for Artificial Intelligence*, 373–382. Springer.

Ross, S.; Gordon, G. J.; and Bagnell, J. A. 2011. No-regret reductions for imitation learning and structured prediction. In *In AISTATS*. Citeseer.

Tîrnăucă, C.; Montaña, J. L.; Ontañón, S.; González, A. J.; and Pardo, L. M. 2016. Behavioral modeling based on probabilistic finite automata: An empirical study. *Sensors* 16(7):958.

Wooldridge, M. 2009. *An introduction to multiagent systems*. John Wiley & Sons.