2017 American Control Conference
Sheraton Seattle Hotel
May 24–26, 2017, Seattle, USA

Adaptiveness and Consistency of Expert Based Learning Algorithms Selecting Reactions to Human Movements

Carol Young[1], Ayesha Khan[1] and Fumin Zhang[1]

Abstract— **Expert based learning algorithms have been used by robots to choose satisfying reactions to human movements. These algorithms often demonstrate random performance that tries to hit a balance between adaptiveness and consistency that matches human's preferences intuitively. This paper provides a rigorous way to quantify the adaptiveness and consistency of the expert based learning algorithms in the context of human robot interaction. It is discovered that a Markov chain model can be used to allow the analysis of both adaptiveness and consistency for several popular expert based learning algorithms. Success of the method has been seen in both simulation and experimental work.**

I. Introduction

The role of robots is not just limited to industry; friendly, safe and portable robots are increasingly being used for everyday household chores as well. For example, the iRobot's Roomba is a popular device that automatically vacuums the house regardless of any obstacles present on the floor. With the increasing penetration of robots in our everyday lives, it has become extremely important that robots learn how to interact with humans in a safe and predictable way [1], [2].

For safe and predictable human-robot interaction (HRI), research is currently being done in order to enable robots make the right decision when collaborating with humans [3]–[5]. One of these methods involves online learning algorithms [6]. An online learning algorithm typically interacts with a concept, which is an unknown mapping of observed parameters, to an expected output. Consider the case with two possible outputs; the expected output with the higher probability of occurring is denoted as the preference and the other as deviation. The algorithm uses the observed parameters to predict the expected output of the concept, and then implant that output as the actual output. Afterwards, a feedback is implemented where the expected output is compared to the actual output from the system. If the expected and actual outputs match, we say the selection is correct, else we say that the learning algorithm has incurred an error. Based on the feedback i.e. whether an error happened or not, the learning algorithm will compute its internal parameters to try to make the correct prediction for the next interaction. Since these online learning algorithms compute updates sequentially, they require less memory and processing time as compared to offline, i.e. batch learning algorithms. Hence, they are ideal for decentralized robotic systems [7]. In this paper, we will focus on expert based learning algorithms that includes the Winnow [8], weighted majority [9], [10] and the dual expert algorithm [6].

One key characteristic of these algorithms is the behavior each algorithm exhibits when it encounters uncertainty in the concept. In HRI, concept can be a specific behavior or reaction of a human. Generally, robustness is used as a measure for tolerance to uncertainty, and has been studied by means of error bounds for various online learning methods [8], [10]. However robustness only measures how many errors will occur, ignoring the sequence of error occurrences. Since predictability is an important factor for HRI [1], consistency, a specialized form of robustness, is needed to measure how often the output of the algorithm changes when there is uncertainty in the concept.

A second key characteristic of these algorithms is adaptiveness. A system is said to be more adaptive if it changes its output quickly when the concept drifts, i.e. changes over time. There are several online learning algorithms designed specifically to adapt to drift, including changing experts [11], regret minimization [12], and average tracking [13].

The above mentioned characteristics indicate how well an algorithm might perform in real life scenarios. For example, consistency is important in HRI systems where the human might occasionally interact with a robot in unpredictable ways or, sensors have noise in measuring the parameters and feedback, but the robot's output needs to remain predictable to the human. On the other hand, adaptiveness is important in HRI systems where the human changes their mind or preference, over time causing drift in the concept. The robot should be able to change its output accordingly without incurring a lot of errors. Hence, there is a trade off between these two characteristics that can be balanced [6].

Even though these parameters are important to gauge the performance of a learning algorithm, to the best of our knowledge, there has been no quantifiable way to measure them. Previous works mostly use error or mistake bounds as an indication of algorithm performance. In order to better understand the underlying dynamics of these algorithms, we propose a Markov chain analysis to quantify learning algorithms.

Markov chains have been used extensively in the literature for a wide range of applications. Some applications include using hidden Markov models for speech recognition, computing high dimensional integrals and analyzing exploration strategies by autonomous agents. In this paper, we propose

*The research work is supported by ONR grants N00014-14-1-0635 and N00014-16-1-2667; NSF grants CMMI-1436284 and OCE-1559475; NRL N0017317-1-G001; and NOAA NA16NOS0120028, and an ARCS Scholar Award

[1]Carol Young, [1]Ayesha Khan and [1]Fumin Zhang are with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, 30332, USA {cyoung44,ayeshakhan,fumin}@gatech.edu.

using Markov chains in order to evaluate the adaptiveness and consistency of learning algorithms. To the best of our knowledge, this is the first time Markov chains are being used to analyze the performance of such algorithms.

In this paper, we consider three different 'two expert' learning algorithms; the Winnow algorithm, the weighted majority algorithm (WMA) and the dual expert algorithm (DEA). We show that each of these simple algorithms can be represented and analyzed as a Markov chain. We then use first step analysis in order to calculate the switching time among various states of these algorithms. We claim that switching time measured from a state with the preferred output to a state, with a deviating output can be used as a measure of consistency. Moreover, the switching time measured from a state with a deviating output to one with the preferred output, can be used as a measure on adaptiveness. Thus for consistency, a longer switching time implies greater consistency and vice versa. Likewise, for adaptiveness, a shorter switching time implies greater adaptiveness and vice versa. We analyze these characteristics for each of the aforementioned algorithms and justify the obtained results. This novel method has the potential to be generalized as a means to gauge consistency and adaptiveness of online learning algorithms.

The layout of the rest of the paper is as follows: Section II introduces two expert learning algorithms. Section III defines Markov chains for WMA and Winnow. Section IV shows the analysis of DEA. Section V compares the obtained results while Section VI is the conclusion.

II. Two-Expert Learning Algorithms

In this paper, we analyze two-expert learning algorithms. We assume a "concept" will produce two outputs with uncertainty, and the learning algorithm tries to select one of two actions to react to one of the outputs. If the choice of action is compatible with the output from the concept, then we say the choice is made correctly. Otherwise, we say an error is made by the algorithm. In the human robot interaction context, the two outputs can be two different human behaviors, and the actions can be two reactions a robot has to choose from. See our previous work [6] for an example where a mobile robot needs to decide whether to move to the left or right, when encountering a human in a corridor. For this example, the two actions will be moving to the left or to the right for the robot, and the two results will be whether the human is blocked by the robot or not. If the robot made a correct choice then the human will not be blocked. Otherwise, the human will be blocked.

In a two-expert learning algorithm, each expert is assigned a weight. The two weights are adjusted after the algorithm made a selection of action, and compared the outcome of that action with the output from the concept. If a correct choice is made, then the weight associated with the action chosen will increase, otherwise the weight will decrease. The algorithm will select the next action by comparing the two weights, and the action associated with the larger weight will always be selected. The algorithm generally starts by assigning equal weights to both actions with a tie breaker rule to select one of the actions. A general outline of a two-expert algorithm is given in Algorithm 1. Line 7 in Algorithm 1 serves as a tie breaker.

Algorithm 1 Two Expert Learning Algorithm

```
1:  Set W_1 = W_2 = 0.5
2:  if W_1 > W_2 then
3:      a = 1
4:  else if W_1 < W_2 then
5:      a = 2
6:  else if W_1 = W_2 then
7:      a = 1
8:  end if
9:  Choose the action associated with W_a
10: Interact with the concept
11: Compare measured outcome with expected outcome
12: if Error then
13:     Decrease weight W_a
14: else if Correct then
15:     Increase weight W_a or maintain the value of W_a
16: end if
```

In order to analyze the performance of the learning algorithms, some assumptions on the uncertainty associated with the concept is necessary. We assume that the probability of action 1 being compatible with the concept is p_1, and the probability for action 2 being compatible with the concept is p_2. We assume that $p_1 + p_2 = 1$ and $p_1 \neq p_2$. If $p_1 > p_2$, then we say that the concept prefers action 1. If $p_1 < p_2$, then we say the concept prefers action 2. We allow the values of p_1 and p_2 to update.

The two-expert learning algorithm will try to learn the preference of the concept. If the preference changes, the learning algorithm will need to adjust to this drift. This requires adaptiveness of the algorithm. On the other hand, if the preference of the concept only changes temporarily, then the learning algorithm should tolerate such changes. This requires consistency of the algorithm. In this paper, we consider three different expert learning algorithms: the Winnow algorithm, the WMA and the DEA. The difference in these algorithms is the way in which the weight of each expert changes after every interaction with the concept. The Winnow algorithm and the WMA are popular two-expert algorithms. The DEA is proposed in our previous work [6].

III. Markov Chains for Two Algorithms

Here we construct Markov chains for the WMA and the Winnow algorithm. The DEA will be discussed in section IV. Let the ratio of weights W_1 and W_2 be defined by R such that $R = \frac{W_1}{W_2}$. We can represent the behaviors of R in the form of a Markov chain for each of the two algorithms.

A. Weighted Majority Algorithm

The WMA either allows a decrease in the weights, or the weights to maintain their previous value after, each selection

is made. In Algorithm 1, line 13 can be replaced by $W_a = \frac{W_a}{2}$ and line 15 by $W_a = W_a$.

At the start of the algorithm, W_1 and W_2 are both initialized to $\frac{1}{2}$ and thus $R = 1$. At this point action 1 is selected. Then if the concept prefers action 1 the selection is correct, thus $W_1 = W_1$, $W_2 = W_2$ and $R = 1$. However if the concept prefers action 2 then $W_1 = \frac{W_1}{2}$, $W_2 = W_2$ and $R = \frac{1}{2}$.

When $R = \frac{1}{2}$ if the concept prefers action 1 the selection is incorrect thus $W_1 = W_1$, $W_2 = \frac{W_2}{2}$ making $R = 1$. However if the concept prefers action 2 then $W_1 = W_1$ $W_2 = W_2$ and $R = \frac{1}{2}$. Thus, R can only take two possible values: $R = \frac{1}{2}$ or $R = 1$ at all times.

Let the states of a Markov chain be defined by the value of R. Then, two states; z_1 and z_2 are defined where z_1 corresponds to $R = 1$ and z_2 corresponds to $R = \frac{1}{2}$. The state transition matrix, $\mathbb{P}_M$ of the WMA, is as follows:

$$\mathbb{P}_M = \begin{bmatrix} p_1 & p_2 \\ p_1 & p_2 \end{bmatrix} \tag{1}$$

Note that the rows correspond to the present states of the Markov Chain and the columns represent next states. For example, if the present state is z_1 then the probability of staying in z_1 is p_1 and the probability of going to z_2 is p_2. The graph of this Markov chain is shown in Fig. 1. This is a typical random switch model. We see that when a transition between z_1 and z_2 happens, a change of selection will be made by the algorithm. We call such change of selection, a **switch**. Then the states z_1 and z_2 are called **switch states.**

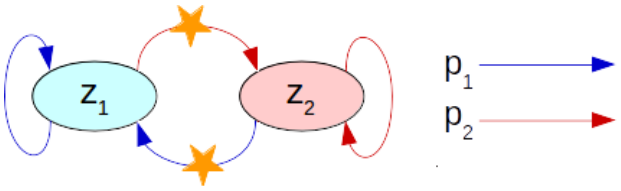

Fig. 1. Markov chain of the WMA. Red arrows represent a probability of p_2, blue of p_1. States that give action 1 are colored light blue and states that give action 2 are colored light red. The orange stars represent the transitions that lead to a switch made by the algorithm.

B. Winnow Algorithm

The Winnow algorithm modifies the WMA by allowing an increase in weights. In algorithm 1, line 13 can be replaced by $W_a = \frac{W_a}{2}$ and line 15 by $W_a = 2W_a$. Hence, the Winnow algorithm either allows a decrease in the weights in response to an error or an unbounded increase in the weights.

As with the WMA, in the Winnow algorithm, both W_1 and W_2 are equal to $\frac{1}{2}$ making the initial $R = 1$. Given any combination of W_1 and W_2, if the concept prefers action 1, there are two possible results; if $W_1 \geq W_2$, then if W_1 is selected and the selection made is correct, $W_1 = 2W_1$. If $W_1 < W_2$ then if W_2 is selected and it is the incorrect choice, then $W_2 = \frac{W_2}{2}$. Since $R = \frac{W_1}{W_2}$, both of these results lead to $R = 2R$. Likewise if the concept prefers action 2 the two possible results are $W_1 = \frac{W_1}{2}$ for an error or $W_2 = 2W_2$. Both of these results lead to $R = \frac{R}{2}$. Therefore all reachable states can be

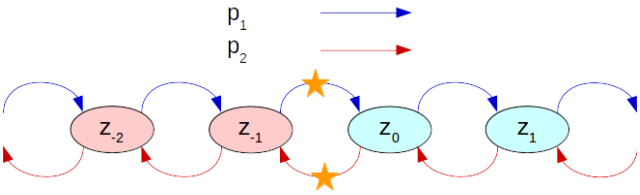

Fig. 2. Markov chain of the Winnow algorithm. Red arrows represent a probability of p_2, blue of p_1. States that give action 1 are colored light blue and states that give action 2 are colored light red. The orange stars represent which transitions would lead to a switch in output.

denoted by $R = 2^m$ where m can be any integer. If we assign z_m to be the state where $R = 2^m$, then the state transition matrix, $\mathbb{P}_W$ is an infinite dimensional matrix as follows:

$$\mathbb{P}_W = \begin{bmatrix} \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \iddots \\ \dots & 0 & p_1 & 0 & 0 & 0 & \dots \\ \dots & p_2 & 0 & p_1 & 0 & 0 & \dots \\ \dots & 0 & p_2 & 0 & p_1 & 0 & \dots \\ \dots & 0 & 0 & p_2 & 0 & p_1 & \dots \\ \dots & 0 & 0 & 0 & p_2 & 0 & \dots \\ \iddots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \tag{2}$$

This is a typical discrete random walk model, with the graph shown in Fig. 2. We can see that the transitions between states z_0 and z_{-1} correspond to a change in choice in the algorithm. Hence the transitions are switches and the two states z_0 and z_{-1} are the switch states.

While the scope of this paper only considers a two expert case, the formulation of these Markov chains can give us insight into the formulation of chains when more experts are used. The relative ratio of weights rather than the weights themselves would still be used, however there would have to be enough ratios so that all the weights could be related to each other. In the worst case for n experts all of the unique states could be represented by using $n-1$ ratios. For Winnow this could become an infinite lattice, while for WMA the chain could be simpler, especially if assumptions were made on the selection of experts. For example, if all experts have an opposing expert that always predicted the opposite, then this could limit the ratio between the maximum and minimum weight, forcing a finite Markov chain.

C. Analyzing the Algorithms

We can analyze the two algorithms modeled by the Markov chains. Suppose the starting state of each algorithm is denoted by z_s. Using first step analysis, it is possible to find the average number of steps that can be taken to reach a particular state from any given state [14]. The results are called **mean hitting times**. We discovered that some of these mean hitting times can be used to measure the adaptiveness and consistency of the learning algorithms.

To increase readability of the analysis, let us suppose the concept always prefers action 1, i.e. $p_1 > p_2$. Note that this does not mean we do not allow change of preference. In fact,

if $p_1 < p_2$, then the analysis is completely symmetric. We first define the measures for adaptiveness and consistency.

a) Weighted Majority Algorithm: For the WMA, its *adaptiveness* is measured by the mean hitting time for the state to start from z_2 and reach z_1. Its *consistency* is measured by the mean hitting time for the state to start from z_1 and reach z_2.

b) Winnow Algorithm: For the Winnow algorithm, its *adaptiveness* is measured by the mean hitting time for the state to start from z_{-1} and reach z_0. Its *consistency* is measured by the mean hitting time for the state to start from z_0 and reach z_{-1}.

Remark 3.1: Adaptiveness is defined as how fast an algorithm is able to change its preference when a drift is encountered. To quantify adaptiveness, when action 1 is preferred, the initial state, z_s should be selected such that action 2 is chosen by the algorithm. A smaller mean hitting time from this initial state to the switch state where action 1 will be chosen, will indicate a better adaptiveness. Consistency is defined as how often an algorithm changes its output in case it encounters a temporary change of preference. If action 1 is preferred, the initial state z_s should be selected such that action 1 is chosen by the algorithm. A larger mean hitting time from this initial sate to the switch state where action 2 will be chosen, will indicate a better consistency

Remark 3.2: For the Winnow algorithm, any state z_s for $s \leq (-1)$ could have been chosen as the initial state to compute adaptiveness, while any state z_s for $s \geq 0$ could have been chose as initial state to compute consistency. However, to compare the Winnow algorithm with the WMA, the states z_{-1} and z_0 are used.

Since both the random switch model and the discrete random walk model are canonical examples to compute mean hitting times in textbooks such as [14], we will just list results here. For the WMA, the measure for adaptiveness is $t_{\mathrm{A}} = \frac{1}{p_1}$, and the measure for consistency is $t_{\mathrm{C}} = \frac{1}{p_2}$. For the Winnow algorithm, the measure for adaptiveness is $t_{\mathrm{A}} = \frac{1}{p_1 - p_2}$, and the measure for consistency is $t_{\mathrm{C}} = \infty$. Comparison between the two algorithms shows that the Winnow is less adaptive, but more consistent, than the WMA. This agrees with our previous simulation results [6].

IV. Dual Expert Algorithm

The DEA was first proposed to balance adaptiveness and consistency[6]. We have observed in both simulation and experiments that the DEA has better adaptiveness than the Winnow algorithm, and better consistency than the WMA. Using the Markov chain analysis method developed in this paper, we are able to give theoretical justification for these observations.

The DEA either allows a decrease in the weights or a *bounded* increase of the weights depending on the previous selection. In Algorithm 1, line 13 is replaced by $W_a = \frac{W_a}{2}$ and line 15 is replaced by

$$W_a = \begin{cases} 2W_a & \text{if } W_a \leq \frac{1}{4} \\ W_a \text{ if } W_a > \frac{1}{4} \end{cases} \tag{3}$$

Note, for ease of analysis we have changed this algorithm slightly from our previous work [6] where the weight W_a was increased by taking its square root after $W_a > \frac{1}{4}$. In this paper W_a is simply held constant when $W_a > \frac{1}{4}$. This change does not affect the performance of the algorithm, but allows much simpler analysis using Markov chains.

Let S be the number of switches that the algorithm has already encountered, then the number of possible states depends on S. We label the states according to the value of S and the ratio R, as illustrated in Figure 3. The state transition matrix $\mathbb{P}_D$ satisfies the following:

$$\mathbb{P}_D = \begin{bmatrix} p_1 & p_2 & 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & p_2 & p_1 & 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & p_1 & p_2 & 0 & 0 & \dots \\ 0 & 0 & p_2 & p_1 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & p_2 & p_1 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & 0 & p_2 & p_1 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \tag{4}$$

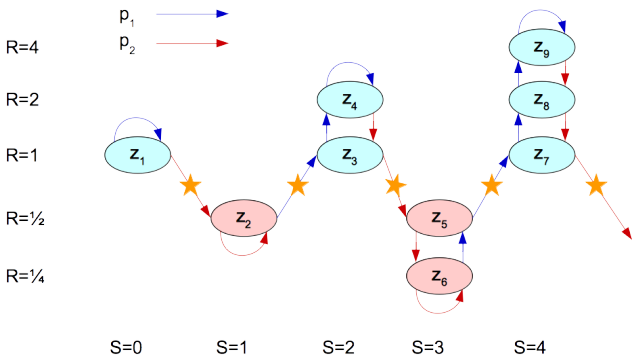

Fig. 3. Red arrows represent a probability of p_2, blue of p_1. States that give action 1 are colored light blue and states that give action 2 are colored light red. The orange stars represent which transitions would lead to a switch in output. S is the number of switches encountered and R represents the ratio of W_1 over W_2

The unique feature of the DEA is, that the switch states depend on S. z_1, z_2, z_3, z_5, z_7 are all switch states. Hence, the mean hitting time, starting and ending at these switch states also depends on the value of S. Between $S = 0$ and $S = 1$, the algorithm behaves like the WMA, but as $S \to \infty$, the algorithm behaves similar to the Winnow algorithm since the number of states increase to infinity.

Given the value of S and taking the assumption that $p_1 > p_2$, we can now analyze the adaptiveness and consistency of the algorithm. Its *adaptiveness* is measured by the mean hitting time from the switching state with $R = \frac{1}{2}$ that corresponds to the value S (see Figure 3), to the next switching state with $R = 1$ that corresponds to the value $S+1$. Its *consistency* is measured by the mean hitting time from the switching state with $R = 1$ that corresponds to the value S, to the next switching state with $R = \frac{1}{2}$ that corresponds to the value $S+1$.

Let t^n_{A} represent the mean hitting time where the starting state is the switching state that corresponds to $R = \frac{1}{2}$ and

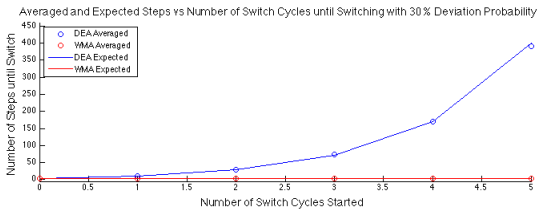

Fig. 4. Expected and averaged steps until switching for a given n with 30% deviation probability

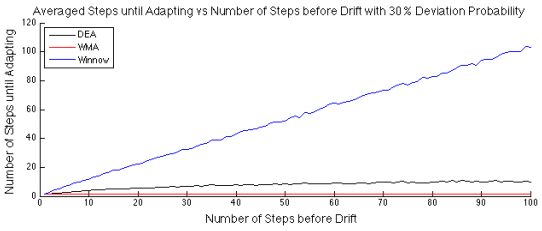

Fig. 5. Averaged steps until adapting for a given number of steps before drift with 30% deviation probability

$S = 2n+1$, and the ending state is the switching state that corresponds to $R = 1$ and $S = 2n+2$. Let t_{C}^n represent the mean hitting time where the starting state is the switching state that corresponds to $R = 1$ and $S = 2n$ and the ending state is the switching state that corresponds to $R = 1$ and $S = 2n+1$. We can compute both values as,

$$t_{\mathrm{C}}^n = \frac{1}{p_2}\frac{1-\left(\frac{p_1}{p_2}\right)^{n+1}}{1-\frac{p_1}{p_2}}$$
$$t_{\mathrm{A}}^n = \frac{1}{p_1}\frac{1-\left(\frac{p_2}{p_1}\right)^{n+1}}{1-\frac{p_2}{p_1}} \quad (5)$$

See the Appendix for more details.

We see that when $n = 0$, $t_{\mathrm{C}}^0 = \frac{1}{p_2}$ and $t_{\mathrm{A}}^0 = \frac{1}{p_1}$. These values are identical to the WMA. When $n \to \infty$, since $p_1 > p_2$, we have $t_{\mathrm{C}}^\infty = \infty$ and $t_{\mathrm{A}}^\infty = \frac{1}{p_1 - p_2}$. These values are identical to the Winnow algorithm. Hence, the WMA and the Winnow algorithms can be seen as extremes of the DEA. For all the finite nonzero values of n, t_{A}^n and t_{C}^n assume values in between the two extremes. This has confirmed our observations from prior simulations and experiments that the DEA is more adaptive than the WMA, and more consistent than the Winnow algorithm [6]. As the number of switches increases, the DEA becomes less adaptive and more consistent.

V. Results

In this section, we simulate WMA, Winnow, and DEA to support the analysis in section III and IV.

We simulated the number of steps until switching takes place for each of the three algorithms; WMA, Winnow and DEA. 200 trials were done, each being 10,000 steps long and having a deviation probability of 30%.

Figure 4 shows the expected and averaged number of steps for both the WMA and the DEA. The Winnow algorithm is not included in the figure, since in this algorithm, the expected number of steps before switching takes place is infinite, and the averaged number of steps is much larger than the calculated steps for WMA and DEA.

As can be seen in Figure 4, WMA has a low consistency irrespective of n, while DEA shows an exponential increase in the number of expected steps, with an increase n, and thus an increase of switches. In addition, it can be seen that the averaged and expected values are close to one another.

Figure 5 shows a simulation for the average number of steps an algorithm encounters, before it adapts to the drift. This is simulated with respect to the number of steps the algorithm encounters, before drift. This comparison was chosen because a larger number of trials before drift occurs could lead to an initial state z_s farther from the switching state, and an increase in n.

The number of steps before drift takes place was ranged from 1 to 100 and 100 trials were simulated for each number of steps. Also, the deviation probability was taken to be 30% both before, and after drift.

Figure 5 shows that for WMA and DEA, the expected number of steps before the algorithm adapts, remains relatively constant, while the Winnow has a constant increase in the averaged number of steps until it adapts.

For WMA, this is because it has a small t_{A} and the initial distance to the switch state is always 1. For the Winnow algorithm, this is because it not only has a greater t_{A} but also the initial distance to the switch state is limited only by the number of trials that have occurred. This shows that as runtime increases, Winnow becomes less adaptive and thus is not ideal for longterm learning.

For DEA, this result shows that the exponential increase of the consistency helps slow the increase of n, limiting both t_{A}^n and the initial distance to the switch state. Therefore, DEA maintains a stable adaptiveness for longterm learning.

Hence, it can be seen that DEA strikes the best balance between consistency and adaptiveness.

VI. Conclusion and Future Work

A novel approach is proposed to measure adaptiveness and consistency for three learning algorithms: the WMA, the Winnow algorithm, and the DEA, when two experts are used. We model the behaviors of the learning algorithms using Markov chains, and discover the connection between mean hitting times, adaptiveness and consistency. Our work is motivated by the HRI experiments in our previous work [6]. Both adaptiveness and consistency are important characteristics for online learning algorithms when applied to human robot interaction. Our work has shown that Markov chain analysis can be used to quantify both characteristics. Future work consists of expanding the Markov chain analysis to the multi-expert cases of learning algorithms.

References

[1] C. Breazeal, "Social interactions in HRI: the robot view," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 34, no. 2, pp. 181–186, 2004.

[2] H. Huttenrauch, A. Green, M. Norman, L. Oestreicher, and K. S. Eklundh, "Involving users in the design of a mobile office robot," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 34, no. 2, pp. 113–124, 2004.
[3] M. Lauckner, F. Kobiela, and D. Manzey, "hey robot, please step back!-exploration of a spatial threshold of comfort for human-mechanoid spatial interaction in a hallway scenario," in *The 23rd IEEE International Symposium on Robot and Human Interactive Communication*. IEEE, 2014, pp. 780–787.
[4] B. Mutlu and J. Forlizzi, "Robots in organizations: the role of workflow, social, and environmental factors in human-robot interaction," in *3rd ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 2008, pp. 287–294.
[5] K. Dautenhahn, S. Woods, C. Kaouri, M. L. Walters, K. L. Koay, and I. Werry, "What is a robot companion-friend, assistant or butler?" in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2005, pp. 1192–1197.
[6] C. Young and F. Zhang, "A learning algorithm to select consistent reactions to human movements," in *2016 American Control Conference (ACC)*. IEEE, 2016, pp. 6152–6157.
[7] L. Kuncheva, "Classifier ensembles for changing environments," in *Multiple Classifier Systems*, ser. Lecture Notes in Computer Science, F. Roli, J. Kittler, and T. Windeatt, Eds. Springer Berlin Heidelberg, 2004, vol. 3077, pp. 1–15.
[8] N. Littlestone and M. K. Warmuth, "The weighted majority algorithm," in *Foundations of Computer Science, 1989., 30th Annual Symposium on*. IEEE, 1989, pp. 256–261.
[9] N. Littlestone, "Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm," *Machine learning*, vol. 2, no. 4, pp. 285–318, 1988.
[10] A. Blum, "Empirical support for winnow and weighted-majority algorithms: Results on a calendar scheduling domain," *Machine Learning*, vol. 26, no. 1, pp. 5–23, 1997.
[11] J. Z. Kolter and M. A. Maloof, "Dynamic weighted majority: An ensemble method for drifting concepts," *The Journal of Machine Learning Research*, vol. 8, pp. 2755–2790, 2007.
[12] K. Crammer, Y. Mansour, E. Even-Dar, and J. W. Vaughan, "Regret minimization with concept drift." in *COLT*, 2010, pp. 168–180.
[13] G. J. Ross, N. M. Adams, D. K. Tasoulis, and D. J. Hand, "Exponentially weighted moving average charts for detecting concept drift," *Pattern Recognition Letters*, vol. 33, no. 2, pp. 191–198, 2012.
[14] N. Privault, "First step analysis," in *Understanding Markov Chains*. Springer, 2013, pp. 95–116.

APPENDIX: MEAN HITTING TIME

Given a Markov chain such as the one in Figure 3, we can compute the mean hitting time from all states in a set Z, to a target state. In Figure 3, each branch corresponding to a value of S can be chosen as the set Z, and the next switch state can be used as the target state.

If P_Z denotes the probability transition matrix of all the states in set Z, then the mean hitting time for each state in Z to hit the target state can be represented by a vector h_A. Then h_A must satisfy,

$$h_A = \mathbf{1} + P_Z h_A \tag{6}$$

where $\mathbf{1}$ is a vector containing all ones. Rearranging equation (6), we can calculate h_A as follows:

$$h_A = (I - P_Z)^{-1}\mathbf{1} \tag{7}$$

where I represents the identity matrix. Then, the mean hitting time from the switch state in set Z can be found as one of the components of the vector h_A.

Next we show how to compute the mean hitting times for the Markov chain of the DEA. Starting from state z_1, we use the branches that correspond to the value of S as the set Z. Then, we get a sequence of transition matrices, P_Z^S. For example,

$$P_Z^{S=0} = \begin{bmatrix} p_1 \end{bmatrix} \tag{8}$$

$$P_Z^{S=1} = \begin{bmatrix} p_2 \end{bmatrix} \tag{9}$$

$$P_Z^{S=2} = \begin{bmatrix} 0 & p_1 \\ p_2 & p_1 \end{bmatrix} \tag{10}$$

$$P_Z^{S=3} = \begin{bmatrix} 0 & p_2 \\ p_1 & p_2 \end{bmatrix} \tag{11}$$

$$P_Z^{S=4} = \begin{bmatrix} 0 & p_1 & 0 \\ p_2 & 0 & p_1 \\ 0 & p_2 & p_1 \end{bmatrix} \tag{12}$$

$$P_Z^{S=5} = \begin{bmatrix} 0 & p_2 & 0 \\ p_1 & 0 & p_2 \\ 0 & p_1 & p_2 \end{bmatrix}. \tag{13}$$

Now let $n = \left\lfloor \frac{S}{2} \right\rfloor$. We can use the n to generalize the matrix $P_Z^{S=2n}$ where Z is selected to contain the branch with $R \geq 1$. We obtain a $n+1$ by $n+1$ matrix having the following structure:

$$P_Z^{S=2n} = \begin{bmatrix} 0 & p_1 & 0 & \dots & 0 & 0 & 0 \\ p_2 & 0 & p_1 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & p_2 & 0 & p_1 \\ 0 & 0 & 0 & \dots & 0 & & p_1 \end{bmatrix} \tag{14}$$

Likewise, if we let the set Z to contain the branch with $R \leq \frac{1}{2}$, we will get $P_Z^{S=2n+1}$ as follows:

$$P_Z^{S=2n+1} = \begin{bmatrix} 0 & p_2 & 0 & \dots & 0 & 0 & 0 \\ p_1 & 0 & p_2 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & p_1 & 0 & p_2 \\ 0 & 0 & 0 & \dots & 0 & p_1 & p_2 \end{bmatrix} \tag{15}$$

It can be seen that matrices $P_Z^{S=2n+1}$ and $P_Z^{S=2n}$ share a very similar structure. The only difference lies in the position of p_1 and p_2 i.e. the locations of p_1 and p_2 are swapped.

Let us assume that $p_1 > p_2$. Let t_{A}^n represents the mean hitting time where the starting state is the switching state that corresponds to $R = \frac{1}{2}$ and $S = 2n+1$, and the ending state be the switching state that corresponds to $R = 1$ and $S = 2n+2$. Let t_{C}^n represents the mean hitting time where the starting state is the switching state that corresponds to $R = 1$ and $S = 2n$ and the ending state as the switching state that corresponds to $R = 1$ and $S = 2n+1$. Then, using equation (7), we can compute both values as follows,

$$\begin{aligned} t_{\mathrm{C}}^n &= \frac{1}{p_2} + \frac{p_1}{p_2^2} + \frac{p_1^2}{p_2^3} + \ldots + \frac{p_1^n}{p_2^{n+1}} = \frac{1}{p_2}\frac{1-\left(\frac{p_1}{p_2}\right)^{n+1}}{1-\frac{p_1}{p_2}} \\ t_{\mathrm{A}}^n &= \frac{1}{p_1} + \frac{p_2}{p_1^2} + \frac{p_2^2}{p_1^3} + \ldots + \frac{p_2^n}{p_1^{n+1}} = \frac{1}{p_1}\frac{1-\left(\frac{p_2}{p_1}\right)^{n+1}}{1-\frac{p_2}{p_1}}. \end{aligned} \tag{16}$$