Proceedings of the Thirtieth International Florida Artificial Intelligence Research Society Conference

Enhancing Multi-Objective Reinforcement Learning with Concept Drift

Frederick C Webber
frederick.webber@us.af.mil
Warfighter Readiness Research Division
Air Force Research Laboratory
2620 Q Street
Wright-Patterson AFB, OH 45433

Gilbert Peterson
gilbert.peterson@afit.edu
Air Force Institute of Technology
2950 Hobson Way
Wright-Patterson AFB, OH 45433

Abstract

Reinforcement learning (RL) is a particular machine learning technique enabling an agent to learn while interacting with its environment. Agents in non-stationary environments are faced with the additional problem of handling concept drift, which is a partially-observable change that modifies the environment without notification. This causes several problems: agents with a decaying exploration fail to adapt while agents capable of adapting may over fit to noise and overwrites previously learned knowledge. These issues are known as the plasticity-stability dilemma and catastrophic forgetting, respectively. Agents in such environments must take steps to mitigate both problems. This work contributes an algorithm that combines a concept drift classifier with multi-objective reinforcement learning (MORL) to produce an unsupervised technique for learning in non-stationary environments, especially in the face of partially observable changes. The algorithm manages the plasticity-stability dilemma by strategically adjusting learning rates and mitigates catastrophic forgetting by systematically storing knowledge and recalling it when it recognizes repeat situations. Results demonstrate that agents using this algorithm outperform agents using an approach that ignores non-stationarity.

Introduction

Traditional reinforcement learning (RL) methods ignore non-stationarity (Berro and Duthen 2001), making them ill-suited for the real world. RL agents outside of controlled laboratory situations face the problem of concept drift (Widmer and Kubat 1996), partially observable changes that invalidate what the agent has learned. For example, hydroelectric dams don't know how much rain is in the forecast, impacting the trade off between retaining drinking water and generating power; elevator controllers aren't informed of meeting schedules, leading to sub-optimal dispatch; and traffic controllers don't know of route closures. RL agents in environments such as these must be able to react to non-stationarity caused by such partially observable changes. Without a method to detect and handle non-stationarity, agents are prone to forget what they learned or not learn at all.

Some existing research implicitly addresses non-stationarity by assuming it doesn't exist. These agents handle mild non-stationarity (Kaelbling, Littman, and Moore 1996), but must continuously relearn a policy (da Silva et al. 2006). After an agent converges to an optimal policy, if changes in the environment force the agent to adjust, an unintended consequence is that the agent forgets the old policy. This is known as *catastrophic forgetting*.

Non-stationary environments also highlight the problem of the plasticity-stability dilemma (Goldberg and Matarić 2003), which is the trade-off based on the learning rate. *Plasticity* is the ability of RL agents to change inherent to RL. Agents with a lower learning rate are cautious not to overreact to noise in the data, thereby giving them *stability*. Accordingly, RL algorithms that implicitly adapt to non-stationary conditions must have a carefully tuned learning rate because low rates prevent adaptation and high rates can result in thrashing or over fitting to noise.

One way to detect non-stationarity is through concept drift detection. Concept drift handles the case when non-stationarity is not directly observable (i.e. not part of the state description) and the agent is unable to force the environment to behave in a certain manner. Past research has explicitly combined concept drift methods with reinforcement learning (Choi, Yeung, and Zhang 2001; Doya et al. 2002; da Silva et al. 2006). These methods only exploit information from a single objective. The algorithm proposed in this paper exploits multiple objectives.

This work presents an algorithm that combines a concept drift classifier with multi-objective reinforcement learning (MORL) to produce an unsupervised technique for learning in non-stationary environments. The algorithm mitigates catastrophic forgetting and manages plasticity and stability by strategically adjusting learning rates. It also provides a testing method for this type of problem. Further, the algorithm does not assume a small number of concepts, nor does it use a model as in prior work (Choi, Yeung, and Zhang 2001; da Silva et al. 2006; Doya et al. 2002).

Preliminary results indicate that a drift-aware agent that handles both catastrophic forgetting and the plasticity-stability dilemma outperforms an agent that does not model drift and thus handles neither situation.

Related Work

Reinforcement learning (RL) enables agents to optimize performance by associating feedback signals with previous actions and observed states (Sutton and Barto 1998). As agents experience their environment, they gain information in the form of the tuple $\langle s, a, r, s' \rangle$ where present state s and future state s' are from the space $\mathcal{S}$, r is a reward in the space of possible rewards $\mathcal{R}$, and actions a are selected from possible choices $\mathcal{A}$. From this experience, reinforcement learning agents learn a behavior policy, π, to optimize the expected reward.

Reinforcement learning algorithms use the Markov assumption that all an agent needs to know is the present state and that the history of how the agent got there is irrelevant. However, in non-stationary environments, the Markov assumption is invalidated, requiring reinforcement learning agents to incorporate additional techniques. Previous RL works that explicitly handle non-stationary environments are prone to catastrophic forgetting and suffer from the plasticity-stability dilemma. The field of concept drift provides a way for an agent to explicitly address non-stationarity.

Concept drift is a data classification learning problem on non-stationary temporally-ordered data involving a hidden context or concept, such that the correct classification of data changes over time (Gama et al. 2014; Zliobaite 2009). Concept drift is relevant in environments where the concept cannot be directly observed, but it can be estimated based on agent history, and when the agent's actions do not impact the concept. Some example concepts include weather prediction (Schlimmer and Granger Jr 1986); a predator identifying a rabbit that changes color each season (Schlimmer and Granger Jr 1986); and consumer interests (Gama et al. 2014).

Reinforcement Learning in Non-stationary Environments

RL agents can implicitly handle small amounts of non-stationarity (Kaelbling, Littman, and Moore 1996) through approximation or by maintaining a sufficient learning rate; however, these approaches can amplify problems or fail to converge. This section reviews works that attempt to explicitly handle non-stationarity.

Dynamic Learning Rates One way to address the plasticity-stability dilemma is to use a changing learning rate (Bowling and Veloso 2002). This provides stability when performance is as expected and plasticity when performance drops. One such method, the Win or Learn Fast (WoLF) approach (Bowling and Veloso 2002), extends iterative gradient ascent to learn a policy. WoLF tracks an 'average' behavior policy $\bar{\pi}$ in addition to the current policy π. When the expected result from $\bar{\pi}$ outperforms the result from π, the agent is not performing as well as anticipated, and so the learning rate increases to δ_{losing}. Otherwise, the agent is doing well and so uses the more conservative $\delta_{winning}$ rate. WoLF uses policy hill climbing (PHC) (Bowling and Veloso 2002), later referred to as *WinOrLearnFast-PHC*.

Explicit Handling of Concept Drift Learning in RL A few have undertaken integrating concept drift with RL (Choi, Yeung, and Zhang 2001; da Silva et al. 2006; Doya et al. 2002). Their algorithms use multiple models that dynamically activate the most relevant policy. Additionally, these approaches for single objective RL require assuming that:

- there are a finite number of concepts (not necessarily a known quantity) that each have distinct dynamics; and
- the concept changes infrequently.

A concept drift-aware reinforcement learning algorithm called RL-CD detects context change using the reward prediction accuracy (da Silva et al. 2006). The equation specifying the instantaneous error in reward prediction accuracy e_m^R for a given model m is

$$e_m^R = 1 - 2\left(\frac{\Delta R_m^2}{R_{max} - R_{min}}\right) \qquad (1)$$

where R_{max} and R_{min} are the maximum and minimum values, respectively, of $\mathcal{R}$ and ΔR_m is the residual between the expected and actual reward, weighted by the number of trips in recent memory. The quality factors are used to compute a total quality for the model. Because this paper only considers reward residuals, e_m^R is considered interchangeable with e_m^R

$$E_m = E_m + \rho\left(e_m - E_m\right) \qquad (2)$$

using $\rho \in (0, 1]$ to weight the impact of new measurements. If no model quality E_m is above a threshold E_{min}, then a new model is created.

The algorithm maintains a set of models, $\mathcal{M}$, which it adds new models to as they are created. This method does not require assuming a baseline number of models and does not restrict the agent based on its spatial locality (da Silva et al. 2006).

Multi-Objective Reinforcement Learning

Agents in the real world can have multiple objectives competing for attention. The problem of multi-objective optimization is to find one or more solutions that minimizes a set of functions. Such problems are multi-objective optimization (MOO) problems. In interesting MOO problems, achieving more optimal performance in one objective happens at the expense of another (Vamplew et al. 2011). Thus, no single strategy can reduce the information to a single objective and be applicable in all circumstances.

One challenge in MOO is how to characterize solutions in a concise manner. Because MOO solutions have multiple objectives, scalar quantifications leave out information about traits of the solution set. However, scalar values are easy to compare, and so a number of *indicators* have been developed to attempt to characterize the quality of a solution concisely.

Indicators that preserve dominance order relations are Pareto compliant (Coello, Lamont, and Van Veldhuizen 2007). That is, if one solution set strictly outperforms another, the superior solution set will always have a lower (better) value than the worse set. One Pareto compliant indicator is the R2 function. The R2 indicator uses a predetermined set of weighted vectors to scalarize a vector of objectives.

The baseline multi-objective reinforcement learning (MORL) algorithm for this research is multi-objective Q-learning as done by (Van Moffaert, Drugan, and Nowé

2013), where expected optimal actions are identified by computing an indicator of the expected q values $\vec{q}$ for the current state and proposed action.

Enhancing MORL with Concept Drift

This section presents MORL augmented with concept drift (Algorithm 1). The proposed algorithm is based on RL-CD, the indicator-based multi-objective Q learning algorithm (Van Moffaert, Drugan, and Nowé 2013), and the WoLF dynamic learn rate.

This new algorithm, CD-MORL, manages a collection of models, one per unique concept. A model consists of a Q table, the policy π, and a concept error signal, E_m. One model is active at a given time. When a concept change is detected, the active model is replaced. If a previous model was recognized by having E greater than E_{min}, it is re-activated. Otherwise, a new model is created. Model recognition and re-initialization mitigates the catastrophic forgetting. To address the plasticity-stability dilemma, the algorithm uses a variable learn rate, as in Win or Learn Fast (Bowling and Veloso 2002).

Detecting Concept Changes in MORL

The concept drift detection adapts RL-CD (da Silva et al. 2006) for multiple objectives, which maintains an error E computed from sequential e_m^R samples. An agent maintains this error signal for all concept models, including its active concept and all stored concepts, and uses this signal both to explicitly detect concept change and to perform concept recognition.

The conversion to multiple objectives is not straight forward. In (da Silva et al. 2006), the goal of the instantaneous quality is to provide an indicator in the range of $[+1, -1]$, with $+1$ being best and -1 being worst prediction quality. If the environment always returns rewards, then a normalized scalarization function may be applied. To maintain generality, the above quality method is rewritten for vector rewards as

$$e_m = 1 - 2\,scalarize\left(\frac{(\Delta R_m)^2}{(R_{max} - R_{min})^2}\right). \quad (3)$$

The formulation is the same for both minimization and maximization objectives. The error signal, Equation 2, forms the basis for all concept drift detection and concept recognition in the algorithm proposed and tested in this work.

When the concept drift learner detects non-stationarity through low model quality because the error signal goes below the threshold E_{min}, the proposed *CD-MORL* algorithm compares the performance residual with all saved models. Using the state and action information and the received reward, the long-term model quality is computed for all models at all time steps. If a match is found - that is, a stored policy has an error E_m that exceeds the quality threshold E_{match} - then that table is restored. Otherwise, a new model is created. Regardless, the current model is archived.

Algorithm 1 CD-MORL

```
Require: $t_{freeze}$
 1: $m_{cur} \leftarrow NewModel()$
 2: $M \leftarrow m_{cur}$
 3: $s \leftarrow s_0$
 4: $t_{sincefreeze} \leftarrow 0$
 5: repeat
 6:     select and take $a$ according to $\pi_{m_{cur}}(s)$, $Q_{cur}(s)$
 7:     observe $s'$, $\vec{r}$
 8:     $t_{sincefreeze} \leftarrow t_{sincefreeze} + 1$
 9:     if $t_{sincefreeze} >= t_{freeze}$ then
10:         $m_{cur} \leftarrow SelectModel(s, a, \vec{r})$
11:         propose $a'$ based on $\pi_{m_{cur}}$
12:     end if
13:     if The model wasn't changed: then
14:         Perform the Q update:
15:         for all objective $o$ do
16:             Update $Q(s, a, o)$
17:         end for
18:         $\pi_{m_{cur}}(s, a) \leftarrow$ WinOrLearnFast-PHC()
19:     else
20:         $t_{sincefreeze} \leftarrow 0$
21:     end if
22: until $s$ is terminal
```

Algorithm 2 SelectModel

```
Require $s$, $a$, $\vec{r}$
for all $m \in M$ do
    $\Delta R_m \leftarrow \mathbf{Q}(s, a) - \vec{r}$
    $e_m \leftarrow 1 - 2scalarize\left(\frac{(\Delta R_m)^2}{(R_{max}-R_{min})^2}\right)$
    $E_m \leftarrow E_m + \rho(e_m - E_m)$ according to Eq 3
end for
$m_{cur} \leftarrow argmax_m(E_m)$
if $E_{m_{cur}} < E_{min}$ then
    $m_{cur} \leftarrow NewModel()$
    $M \leftarrow M \bigcup m_{cur}$
end if
return $m_{cur}$
```

Estimating Convergence

To allow for a policy to stabilize, the Q table needs to receive a 'reasonable' amount of training before it is ready to detect drift. In (Kearns and Singh 1999), the authors establish a formula predicting the number of samples required to have confidence that the Q table has converged. This result assumes an exploration policy that reaches all states. The number of samples required by this method is impractically large.

Instead, in *CD-MORL*, a new concept is not permitted to be replaced until t_{freeze} steps have passed. When a concept is recognized and its model loaded, the concept is not permitted to change again until a few steps have passed to prevent thrashing.

Experimental Design

Testing used a multi-objective MDP generator called MERLIN (Deon 2015) to generate a set of directed graphs which represent concepts. The graph is a random path connecting discrete nodes (states) such that no states are dead ends and all states are reachable. Tests contained 25 states with 4 actions and 3 objectives. Each action is a state transition. Each concept, then, has different outcomes for each action, and the 3 rewards for reaching each state are different.

Measuring MORL Performance in a Non-stationary Environment

In non-stationary environments, an agent that performs well in one concept may perform poorly after the environment changes to another concept (Cruz, González, and Pelta 2011; Nguyen, Yang, and Branke 2012). This makes comparing the performance of agents impossible unless the concept is controlled. In the field of evolutionary dynamic optimization, the *accuracy* measure controls for when the best possible performance differs among concepts. This measure gives, at time t, the performance of an algorithm with respect to the best and worst performance of all tested algorithms. The formulation, as adapted for minimization algorithms is:

$$accuracy^{(t)}_{F,Algorithm} = 1 - \frac{F^{(t)}_{Algorithm} - Min^{(t)}_{F}}{Max^{(t)}_{F} - Min^{(t)}_{F}} \qquad (4)$$

where $F^{(t)}$ is the current reward vector; $Min^{(t)}_F$ is the best observed performance across all algorithms at time t; and $Max^{(t)}_F$ is the worst observed performance across all algorithms at time t. Because of the need to know best and worst observed, this measure must be done offline. The accuracy measure assumes all algorithms are run in the same environment settings. For these experiments, the fitness function F is the R2 indicator of the three objectives, in this case with a single weight vector that averages the three objectives. Normally, the R2 indicator cannot be guaranteed to be Pareto-compliant, meaning that agents achieving better R2 values cannot be guaranteed to have outperformed agents with inferior R2 values. However, in this case the objectives all have the same domain and characteristics. Thus, in this specific case, an agent with a better accuracy of R2 values is guaranteed to outperform an agent with inferior R2 values. While the agent is minimizing on each objective, a greater accuracy value outperforms a lesser accuracy value.

Experiment Design

Experimentation tested three different agent designs and two agent parameterizations under varying environmental conditions.

The control agent design, *MORL*, uses the baseline implementation patterned after (Van Moffaert, Drugan, and Nowé 2013) using the R2 indicator. The control agent never used WoLF and never detected drift. It always used δ_{losing} as the policy learn rate was greater than $\delta_{winning}$.

The first test agent, *CD-MORL init*, tests the CD MORL algorithm with only the recognition and re-initialization component, that is, with the WoLF component disabled. The *CD-MORL* agent detects drift, recognizes old concepts or re-initializes the agent to deal with new ones, and employs WoLF to stabilize the agent.

For the two *CD-MORL* agents, three parameters were modified: the R2 value required to be deemed stable E_{min}, meaning below that the agent is unstable; the R2 value required to recognize a previous concept $E_{match}, -1 \leq E_{match} \leq E_{min} \leq 1$; and the rate at which new information is incorporated $\rho \in (0,\ 1]$. Due to space restrictions, only two parameterizations are illustrated.

Lastly, the environment was tested under several different designs. While each design used the MERLIN settings described above, the graphs were randomly generated. One graph was discarded because it was ill-suited for testing as a dominant state looped back on itself, but otherwise all graphs were taken as-is. For space reasons, only one configuration is shown.

Concepts were on a set rotation. Within an experiment, concepts were active for a fixed length of time ($t_{interval} \in \{500, 1000\}$), though that length of time is varied across experiments. Also, to assist with convergence of the *CD-MORL* algorithms, these algorithms were assigned a freeze time $t_{freeze} \in \{150, 250, 500\}$ where the agent was forbidden from performing concept recognition and re-initialization. This time was fixed within an experiment. The controlled parameters are the other constants: α at 0.1, γ at 0.9, $\delta_{winning}$ at 0.01, and δ_{losing} at 0.1.

Additionally, the random number generator was reset at the start of a set of trials. The reset only happened when changing between drifting and non-drifting, and WoLF and WoLF-less. These parameters were set first, and so the reset only happened four times. Thirty trials were evaluated at each parameter setting.

Results

The *CD-MORL* agents typically outperformed the reference *MORL* agent, with the complete *CD-MORL* agent typically outperforming the *CD-MORL init* agent. This is illustrated in Figures 1, 2, 3, and 4.

Typically, the *CD-MORL* agent had a greater accuracy than the partial *CD-MORL init* agent, and both typically outperformed the plain MORL agent. Results are averaged over 30 runs and smoothed using a moving average window of nine samples. Because of the normalization in the accuracy measure, relative performance within experiments is comparable, but absolute performance across experiments is not. Figure 1 shows the *CD-MORL* agent outperforming both agents, except a little at the beginning and for one concept near the end, when *CD-MORL init* is slightly better. Both *CD-MORL* agents outperform the baseline *MORL* the entire time. However, the agents are sensitive to parameterization, shown in Figure 2. With just E_{match} and ρ changed, the *MORL* agent outperforms the others one third of the time and is comparable for another third. Figure 3 illustrates the impact of changing the freeze time t_{freeze}. Increasing the freeze length appears to cause the *CD-MORL* agents to perform worse than having a shorter freeze length. Though they are generally marginally better than the baseline *MORL*, *MORL* nearly closes the gap in one concept and

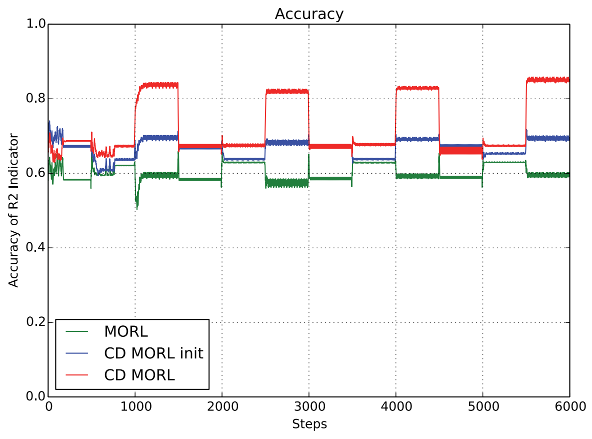

Figure 1: Accuracy of the three agents. Concept duration is 500, $t_{freeze} = 150$, $E_{match} = 0.0$, $\rho = 0.1$.

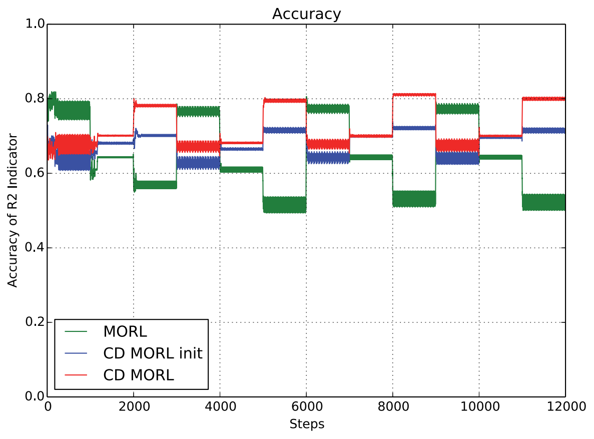

Figure 4: Accuracy of the three agents. Concept duration is changed to 1000, $t_{freeze} = 500$, $E_{match} = 0.0$, $\rho = 0.1$.

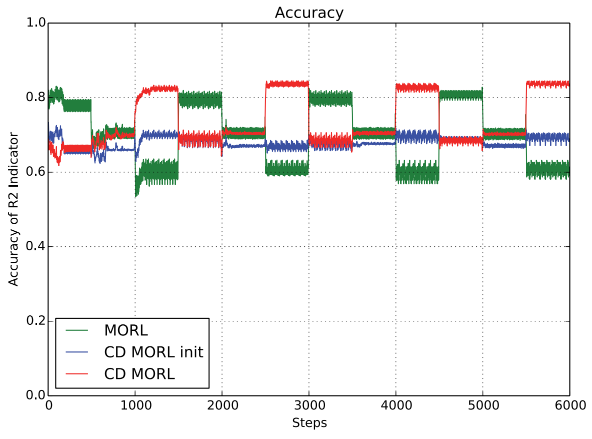

Figure 2: Accuracy of the three agents. Concept duration is 500, $t_{freeze} = 150$, $E_{match} = 0.2$, $\rho = 0.2$. The parameter change adversely impacts the CDMORL algorithms.

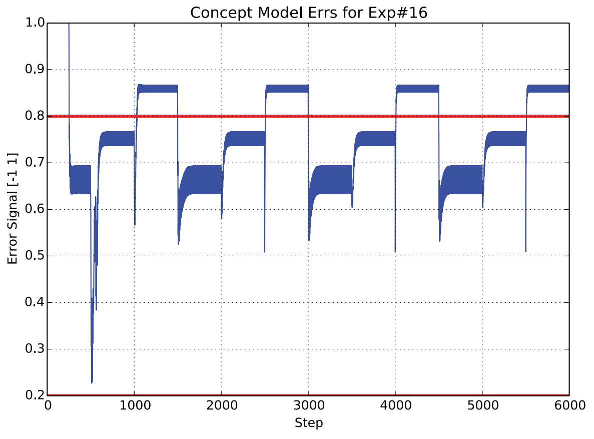

Figure 5: Error signals for one agent. Using $E_{min} = 0.8$ and $E_{match} = 0.2$, no concept change was ever detected. A signal of 1 means no error, -1 means maximum error.

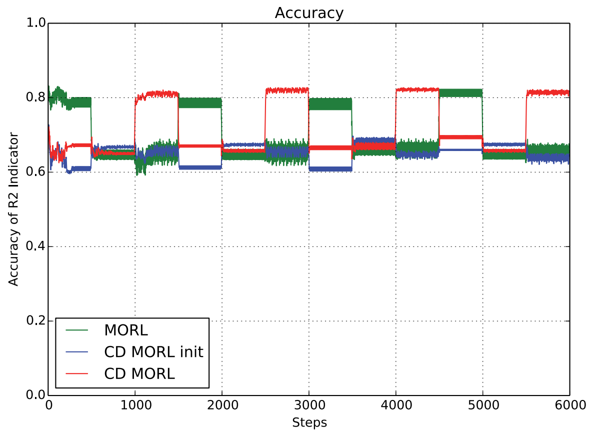

Figure 3: Accuracy of the three agents. Concept duration is 500, $t_{freeze} = 250$, $E_{match} = 0.0$, $\rho = 0.1$. The t_{freeze} parameter adversely impacts the CDMORL algorithms.

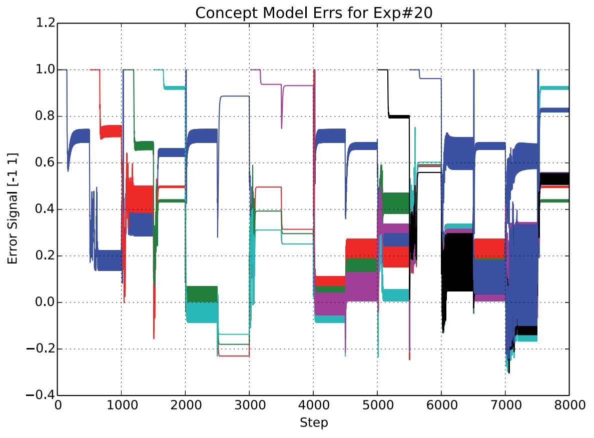

Figure 6: Error signals for one agent. Using $E_{min} = 0.8$ and $E_{match} = 0.2$, seven unique concepts were detected yet none were recognized.

consistently outperforms them on another. Figure 4 shows a longer concept duration and a longer freeze time, where *CD-MORL* outperforms the other two agents two out of three concepts, but *CD-MORL init* performs the same or worse at all times.

The error signal parameters of E_{min}, E_{match}, and ρ have a large impact on classifier performance. Two extreme examples of this are shown in Figures 5 and 6. Each figure shows one instance of an agent, not an average. The error of each unique concept model within that agent is shown starting with its creation step up through the end of the trial. Each unique concept has a unique color (in Figure 5, there is only one concept to see, in Figure 6, blue is used twice). The error signal changes suddenly each time the concept changes, but the scale of the change is dependent on the problem and the parameters. No smoothing was performed on the error signal plots. To illustrate parameter sensitivity, the error plots have two horizontal lines for the thresholds of $E_{min} = 0.2$ and $E_{match} = 0.8$.

Figure 5 shows that the agent often detected drift (any time the signal is below the top red line) and, in this case, never changed concepts (never went below the bottom red line). The agent correctly 'recognizes' the first concept. At the other extreme, as in Figure 6, new concepts were created frequently. Each new concept is created at a true concept change, as the concept changed every 500 steps. However, a correct concept model was not recognized at any point.

Conclusion & Future Work

This work presents an algorithm to address the problem of concept drift, a problem that autonomous agents face in non-stationary environments. By incorporating a concept drift classifier into a multi-objective reinforcement learning agent, it is equipped to mitigate catastrophic forgetting and the plasticity-stability dilemma. Results demonstrate that agents using this algorithm outperform agents that assume a stationary environment.

Future work will examine alternative methods of performing drift detection and recognition, notably by saving more than just a scalarized error signal, since that loses the information advantage provided by multiple objectives. Other work will examine proper ways to react to drift, notably when to store and retrieve concepts and whether it is better to increase or decrease the learning rate under drift.

Acknowledgment

The views expressed in this paper are those of the authors, and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the U.S. Government.

References

Berro, A., and Duthen, Y. 2001. Search for optimum in dynamic environment: a efficient agent-based method. In *Genetic and Evolutionary Computation Conference. Workshop Program*, 51–54.

Bowling, M., and Veloso, M. 2002. Multiagent learning using a variable learning rate. *Artificial Intelligence* 136(2):215–250.

Choi, S. P.; Yeung, D.-Y.; and Zhang, N. L. 2001. Hidden-mode markov decision processes for nonstationary sequential decision making. In *Sequence Learning*. Springer. 264–287.

Coello, C. C.; Lamont, G. B.; and Van Veldhuizen, D. A. 2007. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer.

Cruz, C.; González, J. R.; and Pelta, D. A. 2011. Optimization in dynamic environments: a survey on problems, methods and measures. *Soft Computing* 15(7):1427–1448.

da Silva, B. C.; Basso, E. W.; Bazzan, A. L.; and Engel, P. M. 2006. Dealing with non-stationary environments using context detection. In *Proceedings of the 23rd International Conference on Machine learning*. ACM.

Deon, G. 2015. Random problem generator for multi-task reinforcement learning problems.

Doya, K.; Samejima, K.; Katagiri, K.-i.; and Kawato, M. 2002. Multiple model-based reinforcement learning. *Neural computation* 14(6):1347–1369.

Gama, J.; Žliobaitė, I.; Bifet, A.; Pechenizkiy, M.; and Bouchachia, A. 2014. A survey on concept drift adaptation. *ACM Computing Surveys (CSUR)* 46(4):44.

Goldberg, D., and Matarić, M. J. 2003. Maximizing reward in a non-stationary mobile robot environment. *Autonomous Agents and Multi-Agent Systems* 6(3):287–316.

Kaelbling, L. P.; Littman, M. L.; and Moore, A. W. 1996. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*.

Kearns, M., and Singh, S. 1999. Finite-sample convergence rates for q-learning and indirect algorithms. *Advances in Neural Information Processing Systems*.

Nguyen, T. T.; Yang, S.; and Branke, J. 2012. Evolutionary dynamic optimization: A survey of the state of the art. *Swarm and Evolutionary Computation* 6:1–24.

Schlimmer, J. C., and Granger Jr, R. H. 1986. Incremental learning from noisy data. *Machine Learning* 1(3):317–354.

Sutton, R. S., and Barto, A. G. 1998. *Reinforcement Learning: An Introduction*, volume 1. Cambridge Univ Press.

Vamplew, P.; Dazeley, R.; Berry, A.; Issabekov, R.; and Dekker, E. 2011. Empirical evaluation methods for multiobjective reinforcement learning algorithms. *Machine Learning* 84(1-2):51–80.

Van Moffaert, K.; Drugan, M. M.; and Nowé, A. 2013. Hypervolume-based multi-objective reinforcement learning. In *Evolutionary Multi-Criterion Optimization*, 352–366. Springer.

Widmer, G., and Kubat, M. 1996. Learning in the presence of concept drift and hidden contexts. *Machine Learning* 23(1):69–101.

Zliobaite, I. 2009. Learning under concept drift: an overview. Technical report, Vilnius University.