Autonomous Model Management via Reinforcement Learning - Extended Abstract

Elad Liebman
The Univ. of Texas at Austin
CS Department
eladlieb@cs.utexas.edu

Eric Zavesky
AT&T Research
ezavesky@research.att.com

Peter Stone
The Univ. of Texas at Austin
CS Department
pstone@cs.utexas.edu

ABSTRACT

Concept drift - a change, either sudden or gradual, in the underlying properties of data - is one of the most prevalent challenges to maintaining high-performing learned models over time in autonomous systems. In the face of concept drift, one can hope that the old model is sufficiently representative despite concept drift. Alternatively, one can discard the old data and retrain a new model with (often limited) new data, or use transfer learning to combine the old data with the new to create an updated model. Which of these three options is chosen affects not only near-term decisions, but also future modeling actions. In this abstract, we model response to concept drift as a sequential decision making problem and formally frame it as a Markov Decision Process. Our reinforcement learning approach to the problem shows promising results balancing cost with performance in maintaining model accuracy.

1. INTRODUCTION

As automation grows, more and more industry control systems around us make decisions autonomously, ranging from image understanding [9] to movie recommendation systems [4] and network and service virtualization [1]. Underneath their hoods, many of these systems rely on models, which can be descriptive (capturing the properties of data) or predictive (using known data to predict other, latent properties). Such systems are often susceptible to the nonstationary, ever-changing dynamics of data in the real world. These changes, either gradual or abrupt, are commonly referred to as *concept drift*. Such shifts in the feature distribution and underlying label correspondence constitute a significant challenge to learning systems. In the face of concept drift, we consider the problem of how to generically and adaptively adjust models to mitigate the risks of drift. We refer to this challenge as *model retraining*, or model management. In this abstract we propose a novel *reinforcement learning* (RL) approach, framing the model retraining problem as a sequential decision making task, and harnessing RL concepts to learn a robust policy for model update.

2. RELATED WORK

The issue of concept drift has been studied by multiple researchers in the past 20 years [15, 14, 16, 3]. These works are also connected to the large subfield of online learning [5]. The model retraining problem is also related to the notion of continual, or lifelong learning, both in the context of general machine learning [8], and in RL [10, 7]. However, lifelong learning is not the same as drift adaptation, since in the case of learning under drift, the learner is engaged in the same ongoing task, whereas in lifelong learning, the learner is expected to adapt to new tasks presented sequentially. Both lifelong learning and autonomous model management can also be perceived as part of the transfer learning literature, a rich problem domain studied extensively both in the context of RL [12], and machine learning in general [13, 6]. A key difference between these methods and this abstract is that we do not focus on the specifics of the underlying models or even the data itself. Instead, we propose an RL meta-learner that decides when and how an independent model should be updated.

Appears in: *Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2017), S. Das, E. Durfee, K. Larson, M. Winikoff (eds.), May 8–12, 2017, São Paulo, Brazil.*

3. MODEL RETRAINING AS A MARKOV DECISION PROCESS

Updating the current model of a given system affects not only the ability to act upon the data currently observed, but data observed in the future as well. Update a model too quickly and you may keep getting sidetracked by outlier occurrences, or waste valuable resources. Wait too long to update, and the performance of your system might deteriorate drastically. It therefore makes sense to frame autonomous model management as a sequential decision-making task. As such, this problem is suitably formulated as a Markov Decision Process (MDP) [11]. In this formulation, the system is an agent, which, given the current model, observations of new data, and knowledge of past data, needs to routinely decide whether to update its model, and in what fashion. In our formulation, incoming data modeled by the system can be divided into batches of varying size. For each batch, the system needs to decide how to best process the data. In the process, the agent needs to balance performance and cost (as they are determined for a given domain). We assume that it is unrealistic for an agent to observe the entire batch before making a decision. For this reason, the agent relies on subsampling the data before making a decision. We assume this subsample is unbiased. After observing the subsample, the agent then decides how to best update the model prior to handling the entire set of samples in the batch. This process is repeated in the next timestep indefinitely.

4. LEARNING A POLICY THROUGH APPROXIMATE VALUE ITERATION

Once the model retraining problem is formally defined as an MDP, a suitable policy needs to be learned for this MDP. In certain cases the MDP can be explicitly "solved", resulting in the optimal policy. If the state and action spaces are finite and sufficiently small, this can be done through a dynamic programming procedure called *value iteration* [11]. However, in the case of continuous state spaces this option is no longer feasible. A variety of approximation strategies exist to mitigate this problem. Consider for instance the simplest approach of partitioning a continuous state space via a grid, then applying value iteration on the discretized state space (now rendered discrete and finite). Furthermore, if the state space is Lipschitz-continuous [2], the error induced by this approximation can be bounded, and converges to 0 the tighter the grid becomes.

Let us assume our model can be parameterized as $M = \langle m_1, \ldots, m_k \rangle$. Given a reasonable model for observation distribution, it too can be parameterized, either by its sufficient statistics or using a non-parametric representation $obs_t = \langle d_1, \ldots, d_\rangle$ (examples for such a representation include the weights of an artificial neural net or the coefficients of a regressor). This parameterization lends itself to a straightforward value iteration procedure based on a discretized representation of the parameter space.

5. PROOF OF CONCEPT - DISTRIBUTION MODEL RETRAINING

One type of model maintenance problem that has multiple uses and serves as a good illustration for a framework is that of distribution tracking. Given a constant stream of multidimensional data, we need to generate a model that tracks the properties of the observed data. In this setting, we are interested in modeling the observed data as a multivariate Gaussian, with (μ, Σ) as the sufficient statistics. We consider observations in the current batch $o \in obs_t$ as modeling success cases, or "hits", if they are within a certain factor of our estimated μ. Similarly, we consider it a modeling failure, or a "miss", if an observation is outside that confidence range. At each timestep, the agent is presented with a set of observations obs_t, drawn from some unknown distribution. The agent is then allowed to sample some subset $\widehat{obs_t}$ of the observations, and decide whether it needs to update the model, and how. In this abstract we consider three options at each timestep - keep the model unchanged, retrain a new model from scratch based on the new observations, or use transfer learning to update the old model with the new model.

As a first step, we concretely illustrate our approach using a synthetic domain. This step is useful since controlling the process which generates the data gives us the freedom to both test the validity of our approach and test its limitations. Our synthetic domain is as follows - data is drawn from a 2-dimensional Gaussian distribution with unknown $\langle \mu, \Sigma \rangle$ parameters. At each timestep, a number of observations drawn from a Poisson distribution $|obs_t| \sim \mathbf{Poi}(\lambda)$. Additionally, the distribution shifts in a random walk process at each timestep - μ and Σ drift by factors drawn from a different unknown distribution is added. To make the distribution meaningful, an upper bound is placed on the value of the true underlying variance. We compare our value iteration approach to five baseline policies: (1) a "do nothing" policy which always stays with its current model; (2) a "retrain always" policy, which retrains a model, paying the cost associated with this action, at each timestep; (3) an "adapt always" policy, which always updates the existing model using new data; (4) a random policy, which chooses actions uniformly irrespective of the current state at each timestep; (5) a fixed policy, which adapts the model if the observed data has deviated by more than 25% of the current estimate, and retrains if it has deviated by more than 50%.

We analyze the performance of each model reuse strategy over 30 timesteps with randomly drawn batches in this synthetic domain. The experiment is repeated over 10 iterations at each step to provide a measure of statistical significance to the reported results. The results are provided in Figure 1. As can be observed, using sampled data judiciously, our system significantly outperforms the other baselines. Figure 1(a) shows the overall average reward per step, whereas (b) and (c) show the accuracy and the costs, respectively, illustrating how the AVI approach carefully balances these two considerations in model retraining, reaching equal or better performance while also managing the least cost.

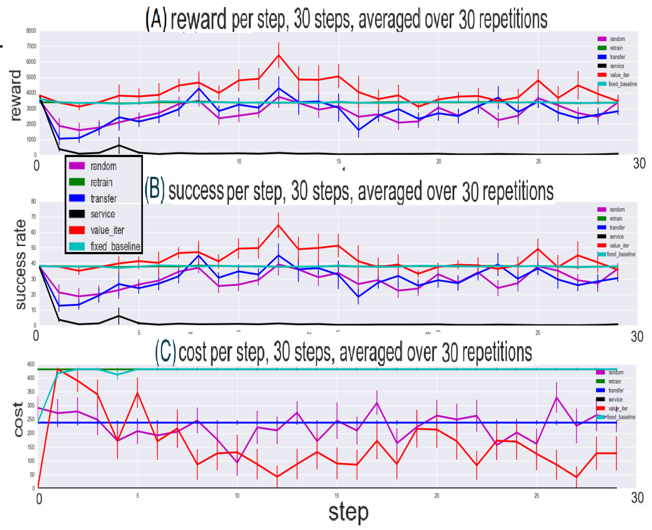

Figure 1: Reward per step over 30 time steps for our approximate value iteration (AVI) system compared to the other model retraining policies. Results are averaged over 30 simulations per step to obtain statistical significance. Results are over the synthetic domain. (a) avg. reward (b) average success rate (c) avg. cost

6. SUMMARY & DISCUSSION

The risk of concept drift has a potentially devastating effect on many real world systems that involve modeling. In this abstract we present a reinforcement learning approach for continual model updating. Rather than building concept drift resistance into the learned model, we frame the model update problem as a sequential decision making task, and learn a policy for when to update the model, and how. This framework is generic and can be easily applied to many different real world systems. We empirically evaluate our approach, and show it outperforms other baseline update policies. Our approach is just the first step toward more robust systems that can adapt to changing environments.

Acknowledgments

This work has taken place in the Learning Agents Research Group (LARG) at UT Austin. LARG research is supported in part by NSF (CNS-1330072, CNS-1305287, IIS-1637736, IIS-1651089), ONR (21C184-01), and AFOSR (FA9550-14-1-0087). Peter Stone serves on the Board of Directors of, Cogitai, Inc. The terms of this arrangement have been reviewed and approved by the University of Texas at Austin in accordance with its policy on objectivity in research.

REFERENCES

[1] M. Chiosi and B. Freeman. AT&T's sdn controller implementation based on opendaylight. Open Daylight Summit, 7 2015.

[2] C.-S. Chow, J. N. Tsitsiklis, et al. An optimal multigrid algorithm for discrete-time stochastic control. 1989.

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

[4] C. A. Gomez-Uribe and N. Hunt. The netflix recommender system: Algorithms, business value, and innovation. *ACM Trans. Manage. Inf. Syst.*, 6(4), Dec. 2015.

[5] J. Kivinen, A. J. Smola, and R. C. Williamson. Online learning with kernels. *IEEE transactions on signal processing*, 52(8):2165–2176, 2004.

[6] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

[7] M. B. Ring. *Continual Learning in Reinforcement Environments*. PhD thesis, University of Texas at Austin, 1994.

[8] P. Ruvolo and E. Eaton. Ella: An efficient lifelong learning algorithm. *ICML (1)*, 28:507–515, 2013.

[9] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 815–823, 2015.

[10] S. Singh, R. L. Lewis, A. G. Barto, and J. Sorg. Intrinsically motivated reinforcement learning: An evolutionary perspective. *IEEE Transactions on Autonomous Mental Development*, 2(2):70–82, 2010.

[11] R. S. Sutton and A. G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998.

[12] M. E. Taylor and P. Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(Jul):1633–1685, 2009.

[13] L. Torrey and J. Shavlik. Transfer learning. *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*, 1:242, 2009.

[14] A. Tsymbal. The problem of concept drift: definitions and related work. *Computer Science Department, Trinity College Dublin*, 106, 2004.

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

[16] I. Žliobaitė. Learning under concept drift: an overview. *arXiv preprint arXiv:1010.4784*, 2010.