Receptivity: Team Description Paper 2018 Fine Tuning of Agent Decision Evaluation

Mike Li

University of Sydney Complex Systems Postgraduate Program, Camperdown NSW 2006, Australia
mili7522@uni.sydney.edu.au

Abstract. Receptivity is a new team for Robocup 2019 in the 2D simulation league. It is based on agent2d-3.1.1 and the recently released Gliders2d. The strategic focus of the team is on improving the evaluation function with machine learning and reinforcement learning techniques. In particular, fine tuning of the agent decision can be made by identifying situations where the intended action was not successfully completed and instead led to negative outcomes such as loss of ball possession. This paper outlines the process of learning and evaluation for decision fine tuning.

1 Introduction

The RoboCup 2D Simulation League abstracts away physical difficulties to focus on building higher level collective intelligence – such as cooperation, team mobility and field control. As such, it demonstrates more advanced tactical decision making than the other Robocup leagues [1]. This is partially due to standardisation of well developed basic functionality, including the agent2d base code released by HELIOS team [2] and subsequently adopted by over 80% of the League's teams [3]. The competitive focus thus shifted towards a great diversity of tactics that can take advantage of chained actions and adapt to different opponents [4–7]. However, such diversity coupled with the heterogenous player types makes effective evaluation of the best action difficult and provides opportunities for improvement of the evaluation algorithm.

Receptivity is a new team for the Robocup 2D soccer simulator [8], written in C++ and based on both agent2d [2] and Gliders2d, which is a recently released extension of agent2d [9]. Gliders2d provides improvements in six targeted areas using the technique of human-based evolutionary computing [10], including action based evaluation and improved team positioning. Other software packages used are:

- librcsc base library
- soccerwindow2 for viewing and debugging games
- Tensorflow and Eigen for learning and evaluation of the neural network

2 Action Selection and Evaluation

Agent actions (for example a pass or dribble to a particular target) are executed over several cycles and composed of simple components, including the kick, turn and dash [11]. Agent2d utilises a two step decision making process for selecting actions: first generating a list of discretised candidate actions and then selecting out of these options using an evaluation function which returns a real value score. Although the basic agent2d evaluation function is quite simple, consisting primarily of measuring the distance to the enemy goal, this can be easily extended to consider multiple components and take account of different situations and opponents. Action dependent evaluation functions were utilised in Gliders2012 [7] and included in Gliders2d [9]. These more complicated evaluation functions can be an important component of improved performance, however they increase the chance that an incorrect assessment is made in certain circumstances. In addition, because of the extended execution time, opponent actions, heterogeneous agent properties, inaccuracies in world modelling and noisy input, actions may not complete as intended. This makes it a use-case for the data mining capabilities of a neural network.

This study develops a method for fine turning the agent decision making process by logging the actions which take place over many games, training a neural network model to detect bad outcomes and integrating this model into the agent code to modify future decisions. The focus here is placed on identifying bad dribbles, which are defined as those which may be intercepted by the opponent before successful completion.

The aim is to implement fine tuning at one level above any existing decision making infrastructure. This hybrid approach does not seek to replace a hand crafted evaluation function which can tuned using a variety of objectives such as maximisation of information flow [12]. These information theoretic measures [13, 14] capture emergent patterns [15–17] which may be difficult to learn based on the experience of each individual agent. Having a hybrid approach makes it different from reinforcement learning methodologies such as the team-partioned, opaque-transition reinforcement learning (TPOT-RL) developed in [18] which learns an appropriate evaluation function from scratch starting from random actions.

3 Data Collection

Each action is treated as a one step episode with the start point at the selection of the action and the end point either when the full action is completed or the action is cancelled. Logging the action and state dependent variables at the start and end of these episodes provides three options:

– Visual exploration of the parameters leading to different outcomes, allowing a greater understanding of potential weakness when facing each specific opponent and highlighting areas that need tactical improvement. These insights may be manually incorporated into improvement to the evaluation function.

– Identification of bad actions as a binary classification task. These, like the bad dribbles defined earlier, are cases when the action did not complete as intended and so should not have been selected by the evaluation function.
– Improved estimation of evaluation score by comparison of the initial real valued evaluation and the outcome at the end of the episode. This is comparable to off policy Q learning [19] which utilises the existing action generator and evaluation function to improve the intractable dimensionality of the action-state space.

4 Exploring Decision Outcomes

We can explore the logged data to find identifiable signatures of bad actions. Figure 1 plots out heatmaps of three attributes logged over 4000 games. When comparing cases of bad dribbles against all dribbles in these attributes a difference can be easily observed. This information can be used by the neural network (which acts as universal function approximators [20]) to classify these bad actions.

5 Performance of Learning Model

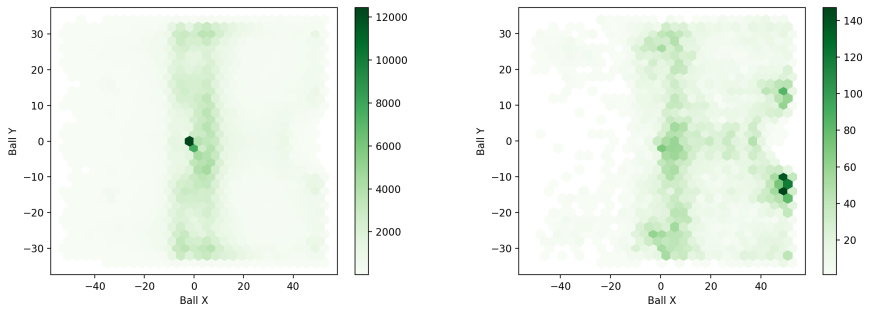
A neural network is learnt offline using Tensorflow in Python and the weights are integrated into the agent code using the C++ matrix library Eigen. Eigen is written entirely within header files, which makes it very simple to link to and compile. This is simpler using Tensorflow in C++, which required either replacing the existing GNU Make process with Bazel (Tensorflow's recommended option) or creating a shared Tensorflow library. The latter was successfully tested by FRA-UNIted [21] and has the benefit of greater flexibility of model architecture and usage.

The current model is able to achieve a test set prediction accuracy of between 90 and 98% and able to improve the goal scoring performance of the team. This accuracy was achieved using a combination of techniques including dropout regularisation [22], batch normalisation [23] and loss functions which are resistant to signal noise [24].

Batch normalisation was found to be effective in reducing overfitting at high epochs. It performs scaling and centring of the output of each layer using the mean and standard deviation of each training batch. Two learnable parameters γ and β allows the network to easily undo this operation if it is beneficial [25]. Using batch normalisation accelerates the training process by reducing internal covariant shift and stabilising the gradient landscape [23].

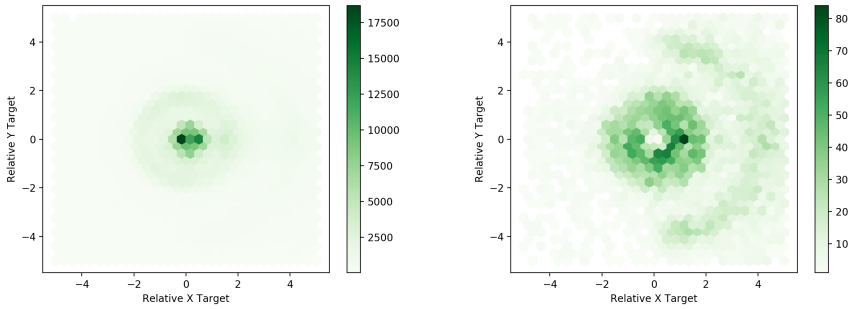
Loss functions which are robust to noise were used because each agent's sensors are inherently noisy due to the properties of the SoccerServer. The cauchy loss function (Eq. 1) and the correntropy loss function (Eq. 2) [24] were tested.

$$\ell(X, Y, h) = \ln\left(1 + \left(\frac{Y - h(X)}{\gamma}\right)^2\right) \tag{1}$$



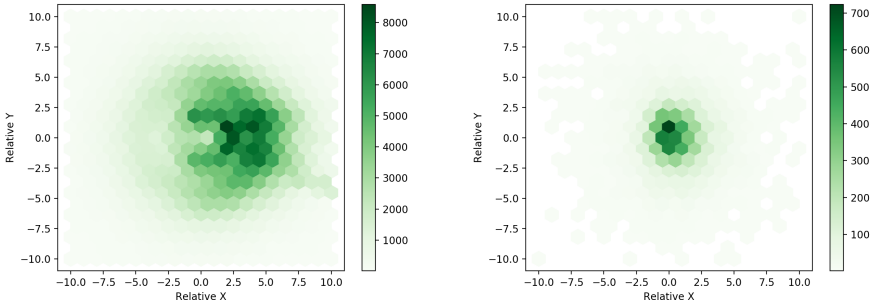
All dribbles Bad dribbles

Current ball position



All dribbles Bad dribbles

Position of target relative to current position



All dribbles Bad dribbles

Position of nearest opponent relative to self

Fig. 1. Heatmaps comparing three attributes in cases of all dribbles versus bad dribbles

$$\ell(X, Y, h) = 1 - \exp\left(- \left(\frac{Y - h(X)}{\sigma} \right)^2 \right) \quad (2)$$

An important consideration when trying to predict bad actions as a binary classification task is the imbalance of training examples. For example, bad dribbles may only occur in 1-5% of dribble cases, so it is difficult to make efficient use of all gathered training data without biasing the prediction outcome. Measures such as the F1 score may also help in evaluating network performance [26].

6 Conclusion and Future Work

Receptivity aims to improve the agent's decision evaluation capabilities using machine learning. For example, by identifying situations where actions are not completed as intended it can fine tuning the decision outcome. While the focus is currently on one step episodes, the future aim will be to extend these look further into the future. There is also still work to be done in finding the best way of using the information from the neural network without disrupting complex multi-step, multi-agent behaviour. In the case of dribbles, some of them may carry high risk of being intercepted by the enemy and classified as a bad action, but these risks may be critical in scoring important goals.

References

1. Budden, D.M., Wang, P., Obst, O., Prokopenko, M.: Robocup simulation leagues: Enabling replicable and robust investigation of complex robotic systems. IEEE Robotics and Automation Magazine **22**(3) (2015) 140–146
2. Akiyama, H.: Agent2D Base Code. https://osdn.net/projects/rctools/releases/p4887 (2010)
3. Prokopenko, M., Wang, P.: Disruptive Innovations in RoboCup 2D Soccer Simulation League: From Cyberoos'98 to Gliders2016. In Behnke, S., Sheh, R., Sariel, S., Lee, D.D., eds.: RoboCup 2016: Robot World Cup XX [Leipzig, Germany, June 30 - July 4, 2016]. Volume 9776 of Lecture Notes in Computer Science., Springer (2017) 529–541
4. Stone, P., Riley, P., Veloso, M.: Defining and using ideal teammate and opponent models. In: Proceedings of the Twelfth Annual Conference on Innovative Applications of Artificial Intelligence. (2000)
5. Prokopenko, M., Wang, P.: Evaluating team performance at the edge of chaos. In Polani, D., Browning, B., Bonarini, A., Yoshida, K., eds.: RoboCup 2003: Robot Soccer World Cup VII. Volume 3020 of Lecture Notes in Computer Science., Springer (2004) 89–101
6. Stone, P., Kuhlmann, G., Taylor, M.E., Liu, Y.: Keepaway soccer: From machine learning testbed to benchmark. In Noda, I., Jacoff, A., Bredenfeld, A., Takahashi, Y., eds.: RoboCup-2005: Robot Soccer World Cup IX. Volume 4020. Springer Verlag, Berlin (2006) 93–105
7. Prokopenko, M., Obst, O., Wang, P., Held, J.: Gliders2012: Tactics with action-dependent evaluation functions. In: RoboCup 2012 Symposium and Competitions: Team Description Papers, Mexico City, Mexico, June 2012. (2012)

8. Chen, M., Dorer, K., Foroughi, E., Heintz, F., Huang, Z., Kapetanakis, S., Kostiadis, K., Kummeneje, J., Murray, J., Noda, I., Obst, O., Riley, P., Steffens, T., Wang, Y., Yin, X.: Users Manual: RoboCup Soccer Server — for Soccer Server Version 7.07 and Later. The RoboCup Federation. (February 2003)
9. Prokopenko, M., Wang, P.: Gliders2d: Source code base for robocup 2d soccer simulation league. CoRR **abs/1812.10202** (2018)
10. Kosorukoff, A.: Human based genetic algorithm. In: Systems, Man, and Cybernetics, 2001 IEEE International Conference on. Volume 5., IEEE (2001) 3464–3469
11. Noda, I., Stone, P.: The RoboCup Soccer Server and CMUnited Clients: Implemented Infrastructure for MAS Research. Autonomous Agents and Multi-Agent Systems **7**(1–2) (July–September 2003) 101–120
12. Cliff, O.M., Lizier, J.T., Wang, X.R., Wang, P., Obst, O., Prokopenko, M.: Quantifying long-range interactions and coherent structure in multi-agent dynamics. Artificial Life **23**(1) (2017) 34–57
13. Schreiber, T.: Measuring information transfer. Physical review letters **85**(2) (2000) 461
14. Lizier, J.T., Prokopenko, M., Zomaya, A.Y.: Local measures of information storage in complex distributed computation. Information Sciences **208** (2012) 39–54
15. Lizier, J.T., Prokopenko, M., Zomaya, A.Y.: A framework for the local information dynamics of distributed computation in complex systems. In Prokopenko, M., ed.: Guided Self-Organization: Inception. Volume 9 of Emergence, Complexity and Computation. Springer Berlin Heidelberg (2014) 115–158
16. Lizier, J.T., Prokopenko, M., Zomaya, A.Y.: Coherent information structure in complex computation. Theory in Biosciences **131** (2012) 193–203
17. Lizier, J.T., Prokopenko, M., Zomaya, A.Y.: Detecting non-trivial computation in complex dynamics. In: European Conference on Artificial Life, Springer (2007) 895–904
18. Stone, P., Veloso, M.M.: Team-partitioned, opaque-transition reinforced learning. In: RoboCup-98: Robot Soccer World Cup II, London, UK, Springer (1999) 261–272
19. Sutton, R.S., Barto, A.G.: Reinforcement learning: An introduction. MIT press (2018)
20. Nielsen, M.A.: Neural networks and deep learning. Volume 25. Determination press USA (2015)
21. Gabel, T., Klppner, P., Godehardt, E.: Fra-united team description 2018. In: RoboCup2018. (2018)
22. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research **15**(1) (2014) 1929–1958
23. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167 (2015)
24. Guan, N., Liu, T., Zhang, Y., Tao, D., Davis, L.S.: Truncated cauchy non-negative matrix factorization for robust subspace learning. IEEE Transactions on Pattern Analysis and Machine Intelligence (2017)
25. Goodfellow, I., Bengio, Y., Courville, A., Bengio, Y.: Deep learning. Volume 1. MIT press Cambridge (2016)
26. Goutte, C., Gaussier, E.: A probabilistic interpretation of precision, recall and f-score, with implication for evaluation. In: European Conference on Information Retrieval, Springer (2005) 345–359