Every Team Deserves a Second Chance: An Interactive 9x9 Go Experience

(Demonstration)

Leandro Soriano Marcolino[1], Vaishnavh Nagarajan[2], Milind Tambe[1]
[1] University of Southern California, Los Angeles, CA, 90089, USA
{sorianom, tambe}@usc.edu
[2] Indian Institute of Technology Madras, Chennai, Tamil Nadu, 600036, India
vaish@cse.iitm.ac.in

ABSTRACT

We show that without using any domain knowledge, we can predict the final performance of a team of voting agents, at any step towards solving a complex problem. This demo allows users to interact with our system, and observe its predictions, while playing 9x9 Go.

Categories and Subject Descriptors

I.2.1 [**Artificial Intelligence**]: Applications and Expert Systems

Keywords

Teamwork; Single and multiagent learning; Social choice

1. INTRODUCTION

It is well known that aggregating the opinions of different agents can lead to a great performance when solving complex problems. For example, voting has been extensively used to improve the performance in machine learning, crowdsourcing, and even board games. Besides, it is an aggregation technique that does not depend on any domain, being very suited for wide applicability. However, a team of voting agents will not always be successful in problem-solving. It is fundamental, therefore, to be able to quickly assess the performance of teams, so that a system operator can take actions to recover the situation in time.

Current multi-agent systems works focus on identifying faulty behavior, or verifying correctness [2, 1]. Such approaches are able to identify if a system is not correct, but provide no help if a correct system of agents is failing to solve a complex problem. Other works focus on team analysis, but require domain knowledge [5, 6].

In this AAMAS conference, we present a novel method to predict the final performance (success or failure) of a team of voting agents, without using any domain knowledge [3]. Hence, our method can be easily applied in a great variety of scenarios. Moreover, our approach can be quickly applied online at any step of the problem-solving process, allowing a system operator to identify when the performance of a team presents issues. This can be fundamental in many applications. For example, consider a complex problem being solved in a cluster of computers. It is undesirable to allocate more resources than necessary, but if we notice that a team is failing in problem solving, we can increase the allocation of resources. Or consider a team playing together a game against an opponent (such as board games, or poker). Different teams might play better against different opponents. Hence, if we notice that a team is having issues, we could dynamically change it. Under time constraints, however, such prediction must be done quickly.

Our approach is based on a prediction model derived from a graphical representation of the problem-solving process, where the final outcome is modeled as a random variable that is influenced by the subsets of agents that agreed together over the actions taken at each step towards solving the problem. Hence, our representation has no dependency on the domain.

In this demo, the user will be able to interact with our system, while playing 9x9 Go games against a team of agents. The agents vote together at each turn of the game, and our prediction system runs in *real time* to show the team's probability of victory. Hence, the user can compare her own estimations with the ones given by our system, and will learn about our work while being entertained by playing a game. Our demo also allows a team to play against an artificial opponent, so that users that do not want to interact with the system can still learn about it and observe its predictions. A video showing our demo is at `http://youtu.be/O3uDQCK9tNs`.

2. PREDICTION METHOD

We briefly present here our prediction methodology. We refer the reader to [3] for a more complete description, and for a theoretical explanation of why the method works. We consider scenarios where agents vote at every step of a complex problem, in order to take common decisions towards problem-solving. Hence, let $\mathbf{T}$ be a set of agents t_i, $\mathbf{A}$ a set of actions a_j and $\mathbf{S}$ a set of world states s_k. The agents vote for an action at each world state, and the team takes the action decided by *plurality voting*. The team obtains a final reward r upon completing all world states. We assume two possible final rewards: "success" (1) or "failure" (0). We define the prediction problem as follows: without using any knowledge of the domain, identify the final reward that will be received by a team. This prediction must be executable at any world state, allowing an operator to take remedy procedures in time.

The main idea of our algorithm is to learn a prediction function, given the frequencies of agreements of all possible agent subsets over the chosen actions. Let $\mathcal{P}(\mathbf{T}) = \{\mathbf{T_1}, \mathbf{T_2}, \ldots\}$ be the power set of the set of agents, a_i the action chosen in world state s_j and $\mathbf{H_j} \subseteq \mathbf{T}$ the subset of agents that agreed on a_i in that world state.

Consider the feature vector $\vec{\mathbf{x}} = (x_1, x_2, \ldots)$ computed at world state s_j, where each dimension (feature) has a one-to-one mapping with $\mathcal{P}(\mathbf{T})$. We define x_i as the *proportion* of times that the chosen

Appears in: *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015), Bordini, Elkind, Weiss, Yolum (eds.), May 4–8, 2015, Istanbul, Turkey.*

action was agreed upon by the subset of agents $\mathbf{T_i}$. That is, $x_i = \sum_{k=1}^{|\mathbf{S_j}|} \frac{\mathbb{I}(\mathbf{H_k}=\mathbf{T_i})}{|\mathbf{S_j}|}$, where $\mathbb{I}$ is the indicator function and $\mathbf{S_j} \subseteq \mathbf{S}$ is the set of world states from s_1 to the current world state s_j.

Hence, given a set $\tilde{\mathbf{X}}$, where for each feature vector $\vec{\mathbf{x_t}} \in \tilde{\mathbf{X}}$ we have the associated reward r_t, we can estimate a function, $\hat{f}$, that returns an estimated reward $\hat{r}$ ($0 \leq \hat{r} \leq 1$) given an input $\vec{\mathbf{x}}$. We classify estimated rewards above 0.5 as "success", and below 0.5 as "failure". In order to *learn* the classification model, we use the features at the final world state, but the learned function can be evaluated at any stage. We use classification by logistic regression, which models $\hat{f}$ as $\hat{f}(\vec{\mathbf{x}}) = \frac{1}{1+e^{-(\alpha+\vec{\boldsymbol{\beta}}^T\vec{\mathbf{x}})}}$.

3. DEMONSTRATION

In this demo we show the predictions given by our methodology in *real time*, in 9x9 Go games. Go is a turn-based board game between two players. At each turn, the players place a stone in an empty intersection of the board. If a group of stones is surrounded by the opponent's stones, they are removed. The stones that surround an area form a territory, whose value is counted by the number of empty intersections inside. The final score is the total amount of territory minus the number of captured stones. A detailed description of the rules can be found in [4]. Go is a very popular game, especially in Asian cultures, and it is currently one of the greatest challenges for Artificial Intelligence.

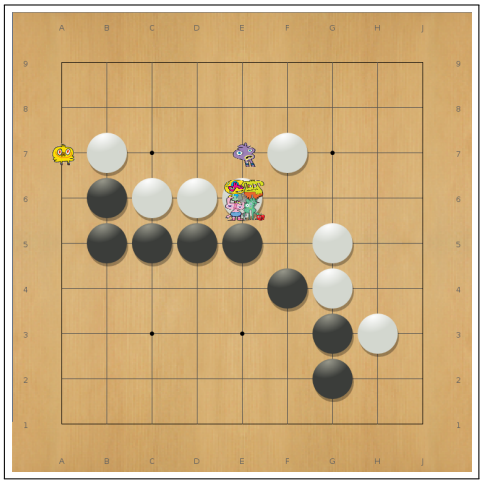
Our system has four different software: Fuego 1.1, GnuGo 3.8, Pachi 9.01, MoGo 4, and two variants of Fuego, in a total of 6 different agents. These are all publicly available Go software. The agents are represented by "cartoon" characters, and the user plays a game in a graphical interface, by clicking in the position where she wants to play. When it is our system's turn, the program displays the votes of all agents in the board, by showing the respective character in the position where the agent voted for (Figure 1(a)).

However, *our demo goes beyond playing a game against a team of agents*. Our main objective is to demonstrate our team performance prediction methodology. In a separate screen (Figure 1(b)), we show in *real time* our estimated probability of the team winning the game. Such predictions are shown in a graph, allowing the user to observe how the probability dynamically changes at each turn. This allows the user to compare our predictions with her own, besides demonstrating that our method can be executed in real time.

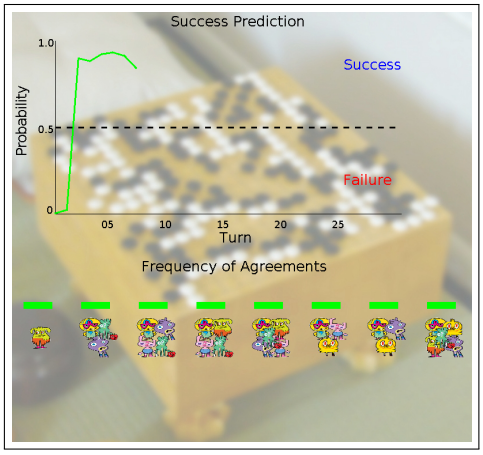
Moreover, we also show the frequency that each subset of agents agreed on the final chosen action. We display the subsets in a clear manner by showing the corresponding characters close together (for additional clarity, we do not display subsets whose frequency is 0 or very close to 0 in comparison to the other subsets). This emphasizes the fact that we use only this information to make a prediction, and that our method is completely domain independent. Moreover, it allows the user to make her own predictions based on the displayed frequencies. Finally, in the end of the game, the system scores the result, and the user is able to verify our predictions.

Since in our work we compare the quality of our predictions for different kinds of teams [3], in this demo we allow two different kinds: "diverse", composed of one copy of each agent; and "uniform", composed of multiple copies of Fuego (with different random seeds). The demo also allows a team to play against an artificial agent (while still showing our predictions). This feature is useful to continually display our demo while no user is interacting with it, in order to attract users to play an actual game and learn about our methodology. We will also have pamphlets to teach the basic rules of Go for users that are unfamiliar with the game.

Acknowledgments: This research was supported by MURI grant W911NF-11-1-0332, and by IUSSTF.



(a) User plays 9x9 Go games against a team of voting agents. Characters display the positions where each agent voted for.



(b) Our predictions are displayed in real time for each turn of the game. We also show the frequency that each subset of agents picks the final chosen action by the team.

Figure 1: In our demo a user plays Go against a team of agents, while observing our real time predictions about the final outcome.

REFERENCES

[1] T. T. Doan, Y. Yao, N. Alechina, and B. Logan. Verifying heterogeneous multi-agent programs. In *AAMAS*, 2014.

[2] E. Khalastchi, M. Kalech, and L. Rokach. A hybrid approach for fault detection in autonomous physical agents. In *AAMAS*, 2014.

[3] V. Nagarajan, L. S. Marcolino, and M. Tambe. Every team deserves a second chance: Identifying when things go wrong. In *AAMAS*, 2015.

[4] Pandanet. Introduction to Go. `http://www.pandanet.co.jp/English/introduction_of_go/`.

[5] T. Raines, M. Tambe, and S. Marsella. Automated assistants to aid humans in understanding team behaviors. In *AGENTS*, 2000.

[6] F. Ramos and H. Ayanegui. Discovering tactical behavior patterns supported by topological structures in soccer-agent domains. In *AAMAS*, 2008.