Every Team Deserves a Second Chance

An extended study on predicting team performance

Leandro Soriano Marcolino[1] · **Aravind S. Lakshminarayanan**[2] · **Vaishnavh Nagarajan**[3] · **Milind Tambe**[4]

Received: date / Accepted: date

Abstract Voting among different agents is a powerful tool in problem solving, and it has been widely applied to improve the performance in finding the correct answer to complex problems. We present a novel benefit of voting, that has not been observed before: we can use the voting patterns to assess the performance of a team and predict their final outcome. This prediction can be executed at any moment during problem-solving and it is completely domain independent. Hence, it can be used to identify when a team is failing, allowing an operator to take remedial procedures (such as changing team members, the voting rule, or increasing the allocation of resources). We present three main theoretical results: (i) we show a theoretical explanation of why our prediction method works; (ii) contrary to what would be expected based on a simpler explanation using classical voting models, we show that we can make accurate predictions irrespective of the strength (i.e., performance) of the teams, and that in fact, the prediction can work better for diverse teams composed of different agents than uniform teams made of copies of the best agent; (iii) we show that the quality of our prediction increases with the size of the action space. We perform extensive experimentation in two different domains: Computer Go and Ensemble Learning. In Computer Go, we obtain high quality predictions about the final outcome of games. We analyze the prediction accuracy for three different teams with different levels of diversity and strength, and show that the prediction works significantly better for a diverse team. Additionally, we show that our method still works well when trained with games against one adversary, but tested with games against another, showing the generality of the learned functions. Moreover, we evaluate four different board sizes, and experimentally confirm better predictions in larger board sizes. We analyze in detail the learned prediction functions, and how they

Leandro Soriano Marcolino
E-mail: l.marcolino@lancaster.ac.uk
Aravind S. Lakshminarayanan
E-mail: aravindsrinivas@gmail.com
Vaishnavh Nagarajan
E-mail: vaishnan@andrew.cmu.edu
Milind Tambe
E-mail: tambe@usc.edu
[1] Lancaster University, Lancaster, Lancashire, LA1 4WA, United Kingdom
[2] Indian Institute of Technology Madras, Chennai, Tamil Nadu, 600036, India
[3] Carnegie Mellon University, Pittsburgh, PA, 15213, USA
[4] University of Southern California, Los Angeles, CA, 90089, USA

change according to each team and action space size. In order to show that our method is domain independent, we also present results in Ensemble Learning, where we make online predictions about the performance of a team of classifiers, while they are voting to classify sets of items. We study a set of classical classification algorithms from machine learning, in a data-set of hand-written digits, and we are able to make high-quality predictions about the final performance of two different teams. Since our approach is domain independent, it can be easily applied to a variety of other domains.[1]

Keywords Teamwork · Collective intelligence · Distributed problem solving · Social choice theory · Single and multiagent learning

1 Introduction

It is well known that aggregating the opinions of different agents can lead to a significant performance improvement when solving complex problems. In particular, voting has been extensively used to improve the performance in machine learning [66], crowdsourcing [55, 4], and even board games [57,64]. Additionally, 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. Moreover, complex problems are generally characterized by a large action space, and hence methods that work well in such situations are of particular interest.

Current works in the multi-agent systems literature focus on identifying faulty or erroneous behavior [44,48,74,13], or verifying correctness of systems [24]. Such approaches are able to identify if a system is not operating correctly, but provide no help if a correct system of agents is failing to solve a complex problem. Other works focus on team analysis. Raines et al. (2000) [68] present a method to automatically analyze the performance of a team. The method, however, only works offline and needs domain knowledge. Other methods for team analysis are heavily tailored for robot-soccer [69] and focus on identifying opponent tactics [60].

In fact, many works in robotics propose monitoring a team by detecting differences in the internal state of the agents (or disagreements), mostly caused by malfunction of the sensors/actuators [42,41,39,40]. In a system of voting agents, however, disagreements are inherent in the coordination process and do not necessarily mean that an erroneous situation has occurred due to such malfunction. Additionally, research in social choice is mostly focused on studying the guarantees of finding the optimal choice given a noise model for the agents and a voting rule [14,49,19], but provide no help in assessing the performance of a team of voting agents.

There are also many recent works presenting methods to analyze and/or make predictions about *human* teams playing sports games. Such works use an enormous amount of data to make predictions about many popular sports, such as American football [67,34], soccer [10,52] and basketball [53,51]. Clearly, however, these works are not applicable to analyzing the performance of a team of voting agents.

Hence, in this paper, we show a novel method to predict the final performance (success or failure) of a team of voting agents, without using any domain knowledge. Therefore, our

[1] This paper is an extended version of Nagarajan et al. (2015) [61]. There are significant differences between this paper and [61], as we listed in detail in the information sheet.

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 team is failing. This can be useful in many applications. For example, consider a complex problem being solved on 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 might wish to 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 predicted to perform poorly, we could dynamically change it. Under time constraints, however, such prediction must be done quickly.

Although related, note that the contribution of this paper *is not* in using a team of experts to solve a problem or make predictions [15] or aggregating multiple classifiers through voting as in ensemble systems [66]. Our objective is, given a team of voting agents, to make a prediction about such a team, in order to estimate whether they will be able to solve a certain problem or not.

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 depends uniquely on the coordination method, and has no dependency on the domain. We explain theoretically why we can make accurate predictions, and we also show the conditions under which we can use a reduced (and scalable) representation. Moreover, our theoretical development allows us to anticipate situations that would not be foreseen by a simple application of classical voting theories. For example, our model indicates that the accuracy can be better for diverse teams composed of different agents than for uniform teams, and that we can make equally accurate predictions for teams that have significant differences in playing strength (which is later confirmed in our experiments). We also study the impact of increasing the action space in the quality of our predictions, and show that we can make better predictions in problems with large action spaces.

We present experimental results in two different domains: Computer Go and Ensemble Learning. In the Computer Go domain, we predict the performance of three different teams of voting agents: a diverse, a uniform, and an intermediate team (with respect to diversity); in four different board sizes, and against two different adversaries. We study the predictions at every turn of the games, and compare with an analysis performed by using an in-depth search. We are able to achieve an accuracy of 71% for a diverse team in 9×9 Go, and of 81% when we increase the action space size to 21×21 Go. For a uniform team, we obtain 62% accuracy in 9×9, and 75% accuracy in 21×21 Go. We also show that we are still able to make high-quality predictions when training our prediction functions in games against one adversary, but testing them in games against another adversary, demonstrating their generality.

We evaluate different classification thresholds using *Receiver Operating Characteristic* (ROC) curves, and compare the performance for different teams and board sizes according to the area under the curves (AUC). We experimentally show in such analysis that: (i) we can effectively make high-quality predictions for all teams and board sizes; (ii) the quality of our predictions is better for the diverse and intermediate teams than uniform (irrespective of their strength), across all thresholds; (iii) the quality of the predictions increases as the board size grows. Moreover, the impact of increasing the action space on the prediction quality occurs earlier in the game for the diverse team than for the uniform team. Finally, we study the learned prediction functions, and how they change across different teams and

different board sizes. Our analysis shows that the functions are not only highly non-trivial, but in fact even open new questions for further study.

In the Ensemble Learning domain, we predict the performance of classifiers that vote to assign labels to set of items. We use the scikit-learn's digits dataset [65], and teams vote to correctly identify hand-written digits. We are also able to obtain high-quality predictions online about the final performance of two different teams of classifiers, showing the applicability of our approach to different domains.

2 Related Work

The research outlined in this article is related to several key areas of related work in multi-agent systems and machine learning, including voting, team performance assessment, (human) sports analytics, agent verification, robotics, multi-agent learning, multiple expert systems and ensemble systems.

We first discuss research in voting. Voting is a technique that can be applied in many different domains, such as: crowdsourcing [55,4], board games [56,57,64], machine learning [66], forecasting systems [37], etc. Voting is very popular since it is highly-parallelizable, easy to implement and provide theoretical guarantees. It is extensively studied in social choice. Normally, it is presented under two different perspectives: as a way to aggregate different opinions, where different voting rules are analyzed to verify if they satisfy a set of axioms that are considered to be important to achieve fairness [63]; or as a way to discover an optimal choice, where different voting rules are analyzed to verify if they converge to always picking the action that has the highest probability of being correct [49,19,14].

Since the classical works of Bartholdi et al. [7,8], many works in social choice also study the computational complexity of computing the winner in elections [3,2], and/or of manipulating the outcome of an election, in general by disguising an agent's true preference [18,23, 79]. There are also works studying the aggregation of partial [77,78] or non-linear rankings (such as pair-wise comparisons among alternatives) [25], since it could be costly/impossible to request agents for a full linear ranking over all possible actions. Some very recent works in social choice also analyze probabilistic voting rules, where the agents' votes affect a probability distribution over outcomes [11,16].

In this work we present a novel perspective to social choice, as instead of studying the computational complexity of manipulation, or the complexity of calculating the winner and the theoretical guarantees of different voting rules, we show here that we can use the voting patterns as a way to *assess the performance* of a team. Such a "side-effect" of voting has not been observed before, and was never explored in social choice theory and/or applications.

Concerning team assessment, the traditional methods rely heavily on tailoring for specific domains. Raines et al. (2000) [68] present a method to build automated assistants for post-hoc, offline team analysis; but domain knowledge is necessary for such assistants. Other methods for team analysis are heavily tailored for robot-soccer, such as Ramos and Ayanegui (2008) [69], that present a method to identify the tactical formation of soccer teams (number of defenders, midfielders, and forwards). Mirchevska et al. (2014) [60] present a domain independent approach, but they are still focused on identifying opponent tactics, not in assessing the current performance of a team.

Team assessment is also closely related to team formation. Team formation is classically studied as the problem of selecting the team with maximum expected utility for a given task, considering a model of the capabilities of each agent [62,31]. Under such framework we could directly compute the expected utility of a team for a certain task. However, for many

domains we do not have a model of the capabilities of the agents, and we would have to rely on observing the team to estimate its performance, as we propose in this work.

There is also a body of work that focuses on analyzing (and predicting the performance of) *human* teams playing sports games. For example, Quenzel and Shea (2014) [67] learn a prediction model for tied NFL American football games, where they use logistic regression to predict the final winner. They study the coefficients of the regression model to determine which factors affect the final outcome with statistical significance. Heiny and Blevins (2011) [34] use discriminant analysis to predict which strategy an American football team will adopt during the game. In soccer, Bialkowski et al. (2014) [10] analyze data from games to automatically identify the roles of each player, and Lucey et al. (2015) [52] also use logistic regression to predict the likelihood of a shot scoring a goal. We can also find examples in basketball. Maheswaran et al. (2012) [53] use a logistic regression model to predict which team will be able to capture the ball in a rebound, while Lucey et al. (2014) [51] study which factors are important when predicting whether a team will be able to perform an open 3-points shot or not.

In the multi-agent systems community, we can see many recent works that study how to identify agents that present faulty behavior [44,48,74]. Other works focus on verifying correct agent implementation [24] or monitoring the violation of norms in an agent system [13]. Some works go beyond the agent-level and verify if the system as a whole conforms to a certain specification [46], or verify properties of an agent system [36]. However, a team can still have a poor performance and fail in solving a problem, even when the individual agents are correctly implemented, no agent presents faulty behavior, and the system as a whole conforms to all specifications.

Sometimes even correct agents might fail to solve a task, especially embodied agents (robots) that could suffer sensing or actuating problems. Kaminka and Tambe (1998) [42] present a method to detect clear failures in an agent team by social comparison (i.e., each agent compares its state with its peers). Such an approach is fundamentally different than this work, as we are detecting a tendency towards failure for a team of voting agents (caused, for example, by simple lack of ability, or processing power, to solve the problem), not a clearly problematic situation that could be caused by imprecision/failure of the sensors or actuators of an agent/robot. Later, Kaminka (2006) [41], and Kalech and Kaminka (2007, 2011) [39, 40] study the detection of failures by identifying disagreement among the agents. In our case, however, disagreements are inherent in the voting process. They are easy to detect but they do not necessarily mean that a team is immediately failing, or that an agent presents faulty behavior/perception of the current state.

This work is also related to multi-agent learning [80], but normally multi-agent learning methods are focused on learning how agents should perform, not on team assessment. An interesting approach has recently been presented in Torrey and Taylor (2013) [76], where they have studied how to teach an agent to behave in a way that will make it achieve a high utility. Besides teaching agents, it should also be possible to teach agent teams. During the process of teaching, it is fundamental to identify when the system is leading towards failure. Hence, our approach could be integrated within a team teaching framework.

Furthermore, the techniques of making predictions by choosing among multiple experts' advice [15], or combining multiple predictions using voting in ensemble systems [66] are also related to this work. Combining multiple forecasters or classifiers has been a very active research area. For example, Sylvester and Chawla (2005) [73] uses a Genetic Algorithm to learn an optimal set of weights when combining multiple classifiers, Chiu and Webb (1998) [17] uses voting to predict the future actions of an agent, and AL-Malaise et al. (2014) [1] use ensembles to predict the performance of a student. Our work, however, is fundamentally

different, as we present a technique to predict the final performance of a team of voting agents. Hence, our contribution *is not* in using a team of agents to make predictions about someone else's performance. We predict the future performance of our own agent team with a domain independent approach, and we do not use ensembles to perform such prediction. We train a single predictor function and run it a single time for each world state where we wish to perform a prediction.

Being able to analyze the performance of a team is, however, also important in the study of ensemble systems. Team formation is also necessary for ensembles, as we should pick the classifiers that lead to the best performance [28]. Hence, the technique introduced in this paper could, in principle, also be applied to make predictions about whether a given ensemble will be successful or not in solving a certain classification task.

Finally, it has recently been shown that diverse teams of voting agents are able to outperform uniform teams composed of copies of the best agent [56,57,38]. Marcolino et al. (2013) [56] show necessary conditions for a diverse team to overcome a uniform team, Marcolino et al. (2014) [57] study how the performance of these teams change as the action space increases, and Jiang et al. (2014) [38] analyze asymptotic guarantees for such teams under classical social choice models, besides studying the performance of ranked voting rules. The importance of having diverse teams has also been explored in the social sciences literature, for example in the works of Hong and Page (2004) [35], and LiCalzi and Surucu (2012) [47], that study models where agents are able to know the utility of the solutions, and the team can simply pick the best solution found by one of its members. Hence, none of these previous works dealt with learning a prediction model to estimate online whether a certain team will win or lose a given game. Additionally, we present here an extra benefit of having diverse teams: we show that we can make better predictions of the final performance for diverse teams than for uniform teams. We will, however, build on the model of Marcolino et al. (2014) when we show that the prediction quality of our technique increases with the action space size.

3 Prediction Method

We start by presenting our prediction method, and in Section 4 we will explain why the method works. We consider scenarios where agents vote at every step (i.e., world state) of a complex problem, in order to take common decisions at every step towards problem-solving. Formally, let $\mathbf{T}$ be a set of agents t_i, $\mathbf{A}$ be a set of actions a_j and $\mathbf{S}$ be a set of world states s_k. The agents must vote for an action at each world state, and the team takes the action decided by the *plurality voting rule*, that picks the action that received the highest number of votes (we assume ties are broken randomly). The team obtains a final reward r upon completing all world states. In this paper, 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 a system operator to take remedial procedures in time.

We now explain our algorithm. The main idea is to learn a prediction function, given the frequencies of agreements of all possible agent subsets over the chosen actions. In order to learn such function, we need to define a feature vector to represent each problem solving instance (for example, a game). Our feature vector records the frequency that each subset of agents was the one that determined the action taken by the team (i.e., the subset whose

action was selected as the action of the team by the plurality voting rule). When learning the prediction function we calculate the feature vector considering the whole history of the problem solving process (for example, from the first to the last turn of a game). When using the learned prediction function to actually perform a prediction, we can simply compute the feature vector considering all history from the first world state to the current one (for example, from the first turn of a game up to the current turn), and use it as input to our learned function. Note that our feature vector does not hold any domain information, and uses solely the *voting patterns* to represent the problem solving instances.

Formally, let $\mathscr{P}(\mathbf{T}) = \{\mathbf{T_1}, \mathbf{T_2}, \ldots\}$ be the power set of the set of agents, a_i be the action chosen in world state s_j and $\mathbf{H_j} \subseteq \mathbf{T}$ be the subset of agents that agreed on a_i in that world state. Consider the feature vector $\mathbf{x} = (x_1, x_2, \ldots)$ computed at world state s_j, where each dimension (feature) has a one-to-one mapping with $\mathscr{P}(\mathbf{T})$. We define x_i as the *proportion* of times that the chosen 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 $\mathbf{X}$ such that for each feature vector $\mathbf{x_t} \in \mathbf{X}$ we have the associated reward r_t, we can estimate a function, $\hat{f}$, that returns an estimated reward between 0 and 1 given an input $\mathbf{x}$. We classify estimated rewards above a certain threshold ϑ (for example, 0.5) as "success", and below it as "failure".

In order to *learn* the classification model, the features are computed at the final world state. That is, the feature vector will record the frequency that each possible subset of agents won the vote, calculated from the first world state to the last one. For each feature vector, we have (for learning) the associated reward: "success" (1) or "failure" (0), accordingly with the final outcome of the problem solving process. In order to *execute* the prediction, the features are computed at the current world state (i.e., all history of the current problem solving process from the first world state to the current one).

We use classification by logistic regression, which models $\hat{f}$ as

$$\hat{f}(\mathbf{x}) = \frac{1}{1 + e^{-(\alpha + \boldsymbol{\beta}^T \mathbf{x})}},$$

where α and $\boldsymbol{\beta}$ are parameters that will be learned given $\mathbf{X}$ and the associated rewards. While training, we eliminate two of the features. The feature corresponding to the subset $\emptyset$ is dropped because an action is chosen only if at least one of the agents voted for it. Also, since the rest of the features sum up to 1, and are hence linearly dependent, we also drop the feature corresponding to all agents agreeing on the chosen action.

We also study a variant of this prediction method, where we use only information about the number of agents that agreed upon the chosen action, but not which agents exactly were involved in the agreement. For that variant, we consider a reduced feature vector $\mathbf{y} = (y_1, y_2, \ldots)$, where we define y_i to be the proportion of times that the chosen action was agreed upon by any subset of i agents:

$$y_i = \sum_{k=1}^{|\mathbf{S_j}|} \frac{\mathbb{I}(|\mathbf{H_k}| = i)}{|\mathbf{S_j}|},$$

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. We compare the two approaches in Section 5.

	Agent 1	Agent 2	Agent 3
Iteration 1	a_1	a_1	a_2
Iteration 2	a_2	a_2	a_1
Iteration 3	a_1	a_2	a_2

Table 1: A simple example of voting profiles after three iterations of problem-solving.

	$\{t_1\}$	$\{t_2\}$	$\{t_3\}$	$\{t_1,t_2\}$	$\{t_1,t_3\}$	$\{t_2,t_3\}$
Iteration 1	0	0	0	1	0	0
Iteration 2	0	0	0	1	0	0
Iteration 3	0	0	0	2/3	0	1/3

Table 2: Example of the full feature vector after three iterations of problem solving.

3.1 Example of Features

We give a simple example to illustrate our proposed feature vectors. Consider a team of three agents: t_1, t_2, t_3. Let's assume three possible actions: a_1, a_2, a_3. Consider that, in three iterations of the problem solving process, the voting profiles were as shown in Table 1, where we show which action each agent voted for at each iteration. Based on plurality voting rule, the action chosen for the respective iterations would be a_1, a_2, and a_2.

We can see an example of how the full feature vector will be defined at each iteration in Table 2, where each column represents a possible subset of the set of agents, and we mark the frequency that each subset agreed on the chosen action. Note that the frequency of the subsets $\{t_1\}$, $\{t_2\}$ and $\{t_3\}$ remains 0 in this example. This happens because we only count the subset where all agents involved in the agreement are present. If there was a situation where, for example, agent t_1 votes for a_1, agent t_2 votes for a_2 and agent t_3 votes for a_3, then we would select one of these agents by random tie braking. After that, we would increase the frequency of the corresponding subset containing only the agent that was chosen (i.e., either $\{t_1\}$, $\{t_2\}$ or $\{t_3\}$).

If the problem has only three iterations in total, we would use the feature vector at the last iteration and the corresponding result ("success" — 1, or "failure" — 0), while learning the function $\hat{f}$ (that is, the feature vectors at Iteration 1 and 2 would be ignored). If, however, we already learned a function $\hat{f}$, then we could use the feature vector at Iteration 1, 2 or 3 as input to $\hat{f}$ to execute a prediction. Note that at Iteration 1 and 2 the output (i.e., the prediction) of $\hat{f}$ will be exactly the same, as the feature vector did not change. At Iteration 3, however, we may have a different output/prediction.

In Table 3, we show an example of the reduced feature vector, where the column headings define the number of agents involved in an agreement over the chosen action. We consider here the same voting profiles as before, shown in Table 1. Note that the reduced representation is much more compact, but we have no way to represent the change in which specific agents were involved in the agreement, from Iteration 2 to Iteration 3. In this case, therefore, we would always have the same prediction after Iteration 1, Iteration 2 and also Iteration 3.

4 Theory

We consider here the view of social choice as a way to estimate a "truth", or the correct (i.e., best) action to perform in a given world state. Hence, we can model each agent as a

	1	2
Iteration 1	0	1
Iteration 2	0	1
Iteration 3	0	1

Table 3: Example of the reduced feature vector after three iterations of problem solving.

probability distribution function (pdf): that is, given the correct outcome, each agent will have a certain probability of voting for the best action, and a certain probability of voting for some incorrect action. These pdfs are not necessarily the same across different world states [56]. Hence, given the voting profile in a certain world state, there will be a probability p of picking the correct choice (for example, by the plurality voting rule).

We will start by developing, in Section 4.1, a simple explanation of why we can use the voting patterns to predict success or failure of a team of voting agents, based on classical voting theories. Such explanation will give an intuitive idea of why we can use the voting patterns to predict success or failure of a team of voting agents, and it can be immediately derived from the classical voting models. However, it fails to explain some of the results in Section 5, and it needs the assumption that plurality is an optimal voting rule. Hence, it is not enough for a deeper understanding of our prediction methodology. Therefore, we will later present, in Section 4.2, our main theoretical model, that provides a better understanding of our results. In particular, based on classical models we would expect to make better predictions for teams that have a greater performance (i.e., likelihood of being correct). Our theory and experiments will show, however, that we can actually make better predictions for teams that are more diverse, even if they have a worse performance. Moreover, we will also be able to build on our model, in Section 4.3, to study the effect of increasing the action space on the prediction quality.

4.1 Classical Voting Model

We start with a simple example to show that we can use the outcome of plurality voting to predict success. Consider a scenario with two agents and two possible actions, a correct and an incorrect one. We assume, for this example, that agents have a probability of 0.6 of voting for the correct action and 0.4 of making a mistake.

If both agents vote for the same action, they are either both correct or both wrong. Hence, the probability of the team being correct is given by $0.6^2/(0.6^2+0.4^2) = 0.69$. Therefore, if the agents agree, the team is more likely correct than wrong. If they vote for different actions, however, one will be correct and the other one wrong. Given that profile, and assuming that we break ties randomly, the team will have a 0.5 probability of being correct. Hence, the team has a higher probability of taking a correct choice when the agents agree than when they disagree ($0.69 > 0.5$). Therefore, if across multiple iterations these agents agree often, the team has a higher probability of being correct across these iterations, and we can predict that the team is going to be successful. If they disagree often, then the probability of being correct across the iterations is lower, and we can predict that the team will not be successful.

More generally than the previous example, we can consider all cases where plurality is the optimal voting rule. In social choice, optimal voting rules are often studied as maximum likelihood estimators (MLE) of the correct choice [19]. That is, each agent is modeled as having a noisy perception of the truth (or correct outcome). Hence, the correct outcome influences how each agent is going to vote, as shown in the model in Figure 1. For example,

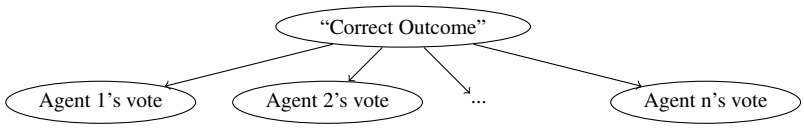


Fig. 1: *"Classical" voting model*. Each agent has a noisy perception of the truth, or correct outcome. Hence, its vote is influenced by the correct outcome.

consider a certain agent t that has a probability 0.6 of voting for the best action, and let's say we are in a certain situation where action a^* is the best action. In this situation agent t will have a probability of 0.6 of voting for a^*.

Therefore, given a voting profile and a noise model (the probability of voting for each action, given the correct outcome) of each agent, we can estimate the likelihood of each action being the best by a simple (albeit computationally expensive) probabilistic inference. Any voting rule is going to be optimal if it corresponds to always picking the action that has the maximum likelihood of being correct (i.e., the action with maximum likelihood of being the best action), according to the assumed noise model of the agents. That is, the output of an optimal voting rule always corresponds to the output of actually computing, by the probabilistic inference method mentioned above, which action has the highest likelihood of being the best one.

If plurality is assumed to be an optimal voting rule, then the action voted by the largest number of agents has the highest probability of being the optimal action. We can expect, therefore, that the higher the number of agents that votes for an action, the higher the probability that the action is the optimal one. Hence, given two different voting profiles with a different number of agreeing agents, we expect that the team has a higher probability of being correct (and, therefore, be successful) in the voting profile where a larger number of agents agree on the chosen action.

We formalize this idea in the following proposition, under the classical assumptions of voting models. Hence, we consider that the agents have a higher probability of voting for the best action than any other action (which makes plurality a MLE voting rule [49]), uniform priors over all actions, and that all agents are identical and independent.

Proposition 1 *The probability that a team is correct increases with the number of agreeing agents m in a voting profile, if plurality is MLE.*

Proof Let a^* be the best action (whose identity we do not know) and $\mathbf{V} = v_1, v_2 \dots v_n$ be the votes of n agents. The probability of any action a being the best action, given the votes of the agents (i.e., $P(a = a^* | v_1, v_2 \dots v_n)$), is governed by the following relation:

$$P(a = a^* | v_1, v_2 \dots v_n) \propto P(v_1, v_2 \dots v_n | a = a^*) P(a = a^*) \tag{1}$$

Let's consider two voting profiles $\mathbf{V}_1, \mathbf{V}_2$, where in one a higher number of agents agree in the chosen action than in the other (i.e., $m_{\mathbf{V}_1} > m_{\mathbf{V}_2}$). Let w_1 be the action with the highest number of votes in $\mathbf{V}_1$, and w_2 the one in $\mathbf{V}_2$.

Without loss of generality (since the order does not matter), let's reorder the voting profiles $\mathbf{V}_1$ and $\mathbf{V}_2$, such that all votes for w_1 are in the beginning of $\mathbf{V}_1$ and all votes for w_2 are in the beginning of $\mathbf{V}_2$. Now, let $\mathbf{V}_1^x$ and $\mathbf{V}_2^x$ be the voting profiles considering only the first x agents (after reordering).

We have that $P(\mathbf{V}_1^{m_{\mathbf{V}_2}} | w_1 = a^*) = P(\mathbf{V}_2^{m_{\mathbf{V}_2}} | w_2 = a^*)$, since up to the first $m_{\mathbf{V}_2}$ agents for both voting profiles we are considering the case where all agents voted for a^*.

Now, let's consider the agents from $m_{\mathbf{V}_2}+1$ to $m_{\mathbf{V}_1}$. In $\mathbf{V}_1$, the voted action of all agents (still w_1) is wired to a^* (by the conditional probability). However, in $\mathbf{V}_2$, the voted action $a \neq w_2$ is not wired to a^* in the conditional probability anymore. As each agent is more likely to vote for a^* than any other action, from $m_{\mathbf{V}_2}+1$ to $m_{\mathbf{V}_1}$, the events in $\mathbf{V}_1$ (an agent voting for a^*) has higher probability than the events in $\mathbf{V}_2$ (an agent voting for an action $a \neq a^*$). Hence, $P(\mathbf{V}_1^{m_{\mathbf{V}_1}}|w_1 = a^*) > P(\mathbf{V}_2^{m_{\mathbf{V}_1}}|w_2 = a^*)$.

Now let's consider the votes after $m_{\mathbf{V}_1}$. In $\mathbf{V}_1$ there are no more votes for w_1, and in $\mathbf{V}_2$ there are no more votes for w_2. Hence, all the subsequent votes are not wired to any ranking (as we only wire $w_1 = a^*$ and $w_2 = a^*$ in the conditional probabilities). Therefore, each vote can be assigned to any ranking position that is not the first. Since the agents are independent, any sequence of votes will thus be as likely. Hence, $P(\mathbf{V}_1|w_1 = a^*) > P(\mathbf{V}_2|w_2 = a^*)$.

Since we assume uniform priors, it follows that:

$$P(w_1 = a^*|\mathbf{V}_1) > P(w_2 = a^*|\mathbf{V}_2)$$

Therefore, the team is more likely correct in profiles where a higher number of agents agree. □

Hence, if across multiple voting iterations, a higher number of agents agree often, we can predict that the team is going to be successful. If they disagree a lot, we can expect that they are wrong in most of the voting iterations, and we can predict that the team is going to fail.

In the next observation we show that we can increase the prediction accuracy by knowing not only how many agents agreed, but also which specific agents were involved in the agreement. Basically, we show that the probability of a team being correct depends on the agents involved in the agreement. Therefore, if we know that the best agents are involved in an agreement, we can be more certain of a team's success. This observation motivates the use of the full feature vector, instead of the reduced one.

Observation 1 *Given two profiles $\boldsymbol{V}_1, \boldsymbol{V}_2$ with the same number of agreeing agents m, the probability that a team is correct is not necessarily equal for the two profiles.*

We can easily show this with an example (that is, we only need one example where the probability is not equal to show that it will not always be equal). Consider a problem with two actions. Consider a team of three agents, where t_1 and t_2 have a probability of 0.8 of being correct, while t_3 has a probability of 0.6 of being correct. As the probability of picking the correct action is the highest for all agents, the action chosen by the majority of the agents has the highest probability of being correct (that is, we are still covering a case where plurality is MLE).

However, when only t_1 and t_2 agree, the probability that the team is correct is given by: $0.8^2 \times 0.4/(0.8^2 \times 0.4 + 0.2^2 \times 0.6) = 0.91$. When only t_2 and t_3 agree, the probability that the team is correct is given by: $0.8 \times 0.6 \times 0.2/(0.8 \times 0.6 \times 0.2 + 0.2 \times 0.4 \times 0.8) = 0.59$. Hence, the probability that the team is correct is higher when t_1 and t_2 agree than when t_2 and t_3 agree.

However, based solely on the classical voting models, one would expect that given two different teams, the predictions would be more accurate for the one that has greater performance (i.e., likelihood of being correct), as we formalize in the following proposition. Therefore, this model fails to explain our experimental results (as we will show later). We will use the term *strength* to refer to a team's performance.

Proposition 2 *Under the classical voting models, given two different teams, one can expect to make better predictions for the strongest one.*

Proof Sketch Under the classical voting models, assuming the agents have a noise model such that plurality is a MLE, we have that the best team will have a greater probability of being correct given a voting profile where m agents agree than a worse team with the same amount of m agreeing agents.

Hence, the probability of the best team being correct will be closer to 1 in comparison with the probability of the worse team being correct. The closer the probability of success is to 1, the easier it is to make predictions. Consider a Bernoulli trial with probability of success $p \approx 1$. In the learning phase, we will see many successes accordingly. In the testing phase, we will predict the majority of the two for every trial, and we will go wrong only with probability $|1-p| \approx 0$.

Of course, we could also have an extremely weak team, that is wrong most of the time. For such a team, it would also be easy to predict that the probability of success is close to 0. Notice, however, that we are assuming here the classical voting models, where plurality is a MLE. In such models, the agents must play "reasonably well": classically they are assumed to have either a probability of being correct greater than 0.5 or the probability of voting for the best action is the highest one in their pdf [49]. Otherwise, plurality is not going to be a MLE. □

Consider, however, that the strongest team is composed of copies of the best agent (which would often be the case, under the classical assumptions). We actually have that, in fact, such agents will not necessarily have noise models (pdfs) where the best action has the highest probability in all world states. In some world states, a suboptimal action could have the highest probability, making the agents agree on the same mistakes [56, 38]. Therefore, when plurality is not actually a MLE in all world states, we have that Proposition 1 will not hold in the world states where this happens. Hence, we will predict that the team made a correct choice, when actually the team was wrong, causing problems in our accuracy. We give more details in the next section.

4.2 Main Theoretical Model

We now present our main theory, that holds irrespective of plurality being an optimal voting rule (MLE) or not. Again, we consider agents voting across multiple world states. We assume that all iterations equally influence the final outcome, and that they are all independent.

Let the final reward of the team be defined by a random variable W, and let the number of world states be S. We model the problem solving process by the graphical model in Figure 2, where $\mathbf{H}_j$ represents the subset of agents that agreed on the chosen action at world state s_j. That is, we assume that the subset of agents that decided the action taken by the team at each world state of the problem solving process will determine whether the team will be successful or not in the end. A specific problem (for example, Go games where the next state will depend on the action taken in the current one) would call for more complex models to be completely represented. Our model is a simplification of the problem solving process, abstracting away the details of specific problems.

For any subset $\mathbf{H}$, let $P(\mathbf{H})$ be the probability that the chosen action was correct given the subset of agreeing agents. If the correct action is a^*, $P(\mathbf{H})$ is equivalent to:

$$\frac{P(\forall t \in \mathbf{H}, t \text{ chooses } a^*, \quad \forall t \notin \mathbf{H}, t \text{ chooses } a \neq a^*)}{P(\exists a' \quad \forall t \in \mathbf{H}, t \text{ chooses } a', \quad \forall t \notin \mathbf{H}, t \text{ chooses } a \neq a')},$$

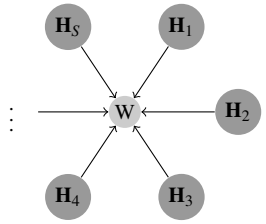


Fig. 2: *Our main model*. We assume that the subset of agents that decided the action of the team at each world state $\mathbf{H}_i$ determines whether the team will be successful or not (W).

where $\mathbf{H}$ is the subset of agents which voted for the action taken by the team.

Note that $P(\mathbf{H})$ depends on both the team and the world state. However, we marginalize the probabilities to produce a value that is an average over all world states. We consider that, for a team to be successful, there is a threshold δ such that:

$$\left\{\prod_{j=1}^{S} P(\mathbf{H}_j)\right\}^{1/S} > \delta \tag{2}$$

We use the exponent $1/S$ in order to maintain a uniform scale across all problems. Each problem might have a different number of world states; and for one with many world states, it is likely that the incurred product of probabilities is sufficiently low to fail the above test, independent of the actual subsets of agents that agreed upon the chosen actions. However, the final reward is not dependent on the number of world states.

We can show, then, that we can use a linear classification model (such as logistic regression) that is equivalent to Equation 2, to predict the final reward of a team.

Theorem 1 *Given the model in Equation 2, the final outcome of a team can be predicted by a linear model over agreement frequencies.*

Proof Getting the log in both sides of Equation 2, we have:

$$\sum_{j=1}^{S} \frac{1}{S} \log(P(\mathbf{H}_j)) > \log(\delta)$$

The sum over the steps (world states) of the problem-solving process can be transformed to a sum over all possible subset of agents that can be encountered, $\mathscr{P}$:

$$\sum_{\mathbf{H}\in\mathscr{P}} \frac{n_{\mathbf{H}}}{S} \log(P(\mathbf{H})) > \log(\delta), \tag{3}$$

where $n_{\mathbf{H}}$ is the number of times the subset of agreeing agents $\mathbf{H}$ was encountered during problem solving. Hence, $\frac{n_{\mathbf{H}}}{S}$ is the frequency of seeing the subset $\mathbf{H}$, which we will denote by $f_{\mathbf{H}}$.

Recall that $\mathbf{T}$ is the set of all agents. Hence, $f_{\mathbf{T}}$ (which is the frequency of all agents agreeing on the same action), is equal to $1 - \sum_{\mathbf{H}\in\mathscr{P}\setminus\{\mathbf{T}\}} f_{\mathbf{H}}$. Also, note that $n_{\emptyset} = 0$, since at least one agent must pick the chosen action. Equation 3 can, hence, be rewritten as:

$$f_{\mathbf{T}} \log(P(\mathbf{T})) + \sum_{\mathbf{H}\in\mathscr{P}\setminus\mathbf{T}} f_{\mathbf{H}} \log(P(\mathbf{H})) > \log(\delta)$$

$$\left(1 - \sum_{\mathbf{H} \in \mathscr{P} \setminus \{\mathbf{T}\}} f_{\mathbf{H}}\right) \log(P(\mathbf{T})) + \sum_{\mathbf{H} \in \mathscr{P} \setminus \mathbf{T}} f_{\mathbf{H}} \log(P(\mathbf{H})) > \log(\delta)$$

$$\log(P(\mathbf{T})) + \sum_{\mathbf{H} \in \mathscr{P} \setminus \mathbf{T}} f_{\mathbf{H}} \log\left(\frac{P(\mathbf{H})}{P(\mathbf{T})}\right) > \log(\delta)$$

Hence, our final model will be:

$$\sum_{\mathbf{H} \in \mathscr{P} \setminus \mathbf{T}} \log\left(\frac{P(\mathbf{H})}{P(\mathbf{T})}\right) f_{\mathbf{H}} > \log\left(\frac{\delta}{P(\mathbf{T})}\right) \tag{4}$$

Note that $\log(\frac{\delta}{P(\mathbf{T})})$ and the "coefficients" $\log(\frac{P(\mathbf{H})}{P(\mathbf{T})})$ are all *constants* with respect to a given team, as we have discussed earlier. Considering the set of all $f_{\mathbf{H}}$ (for each possible subset of agreeing agents $\mathbf{H}$) to be the characteristic features of a single problem, the coefficients can now be *learned* from training data that contains many problems represented using these features. Further, the outcome of a team can be estimated through a linear model. □

The number of constants is exponential, however, as the size of the team grows. Therefore, in the following corollary, we show that (under some conditions) we can approximate well the prediction with a reduced feature vector that grows linearly. In order to differentiate different possible subsets, we will denote by $\mathbf{H}^i$ a certain subset $\in \mathscr{P}$, and by $|\mathbf{H}^i|$ the size of that subset (i.e., the number of agents that agree on the chosen action).

Corollary 1 *If $P(\mathbf{H}^i) \approx P(\mathbf{H}^j)$ $\forall \mathbf{H}^i, \mathbf{H}^j$ such that $|\mathbf{H}^i| = |\mathbf{H}^j|$, we can approximate the prediction with a reduced feature vector, that grows linearly with the number of agents. Furthermore, in a uniform team the reduced representation is equal to the full representation.*

Proof By the assumption of the corollary, there is a $P_{\mathbf{H}^n}$, defined as $P_{\mathbf{H}^n} \approx P(\mathbf{H}^j)$, $\forall \mathbf{H}^j$ such that $|\mathbf{H}^j| = n$. Let $f_n = \sum f_{\mathbf{H}^j}$, over all $|\mathbf{H}^j| = n$. Also, let $\mathbf{N}'$ be the set of all integers $0 < x < N$, where N is the number of agents. We thus have that:

$$\sum_{\mathbf{H} \in \mathscr{P} \setminus \mathbf{T}} f_{\mathbf{H}} \log\left(\frac{P(\mathbf{H})}{P(\mathbf{T})}\right) \approx \sum_{x \in \mathbf{N}'} f_x \log\left(\frac{P_{\mathbf{H}^n}}{P(\mathbf{T})}\right) \tag{5}$$

As $P_{\mathbf{H}^n}$ depends only on the number of agents, we have that such representation grows linearly with the size of the team.

Moreover, note that for a team made of copies of the same agent, we have that $P_{\mathbf{H}^n} = P(\mathbf{H}^j)$, $\forall \mathbf{H}^j$ such that $|\mathbf{H}^j| = n$. Hence, the left hand side of Equation 5 is going to be equal to the right hand side. □

Also, notice that our model does not need any assumptions about plurality being an optimal voting rule (MLE). In fact, there are no assumptions about the voting rule at all. Hence, Proposition 1, Observation 1 and Proposition 2 do not apply, and we can still make accurate predictions irrespective of the performance of a team.

We can also note that the accuracy of the predictions is not going to be the same across different teams, and we may actually be able to have better predictions for a lower performing team. The *learned* constants $\hat{c}_{\mathbf{H}} \approx \log(\frac{P(\mathbf{H})}{P(\mathbf{T})})$ represent the marginal probabilities across all world states. However, Marcolino et al. (2013) [56] and Jiang et al. (2014) [38] show that

Agent	State 1	State 2	State 3
Agent 1	a_1	a_1	a_2
Agent 2	a_1	a_1	a_2
Agent 3	a_1	a_1	a_2
Agent 4	a_1	a_1	a_2

(a) Uniform

Agent	State 1	State 2	State 3
Agent 1	a_1	a_1	a_2
Agent 2	a_2	a_2	a_1
Agent 3	a_3	a_1	a_3
Agent 4	a_3	a_1	a_2

(b) Diverse

Table 4: An example of a diverse and a uniform team, where we are able to make better predictions for the diverse team.

the agent with the highest marginal probability of voting for the correct choice will not necessarily have the highest probability of being correct at all world states. In such world states, the agents tend to agree over the incorrect action that has the highest probability, which will cause problems in our accuracy. Hence, in the following observation, we show that it is possible to make better predictions for a diverse team that has a lower playing performance than a uniform team made of copies of the best agent.

Observation 2 *There exists one diverse and one uniform team (made of copies of the best agent), such that the diverse team leads to better prediction accuracy, even though it has a worse playing performance.*

We can easily show by an example. Let's consider the teams shown in Table 4, where each agent has a probability 1 of voting for the action shown in the cell, for each world state. We consider that the team will be successful when picking a_1, and will fail otherwise. We assume that one problem solving instance will be consisted by a single world state, which may be either State 1, State 2 or State 3, each with a $1/3$ probability. We also consider that the teams play by using plurality voting.

Let's first discuss the uniform team (Table 4 (a)). Given enough training samples, the best predictor that can be created for such team is to predict success when all four agents agree. However, such predictor will be correct only in $2/3$ of the problem instances, as it will fail in problems consisting of State 3. We can also notice that the uniform team will be successful in $2/3$ of the problem instances.

Let's consider now the diverse team in Table 4 (b). With enough training samples, the best predictor that can be learned is to predict success when three agents agree, and failure otherwise. This predictor will be correct in all problem instances, even though the team itself will only be successful in $1/3$ of the problem instances.

Although we only give here an existence result, in Section 5 we experimentally show a statistically significant higher accuracy for the predictions for a *diverse* team than for a *uniform* team, even though they have similar strength (i.e., performance in terms of winning rates). We are able to achieve a better prediction for *diverse* teams both in the end of the problem-solving process and also while doing online predictions at any world state.

4.3 Action Space Size

We present now our study concerning the quality of the predictions over large action space sizes. In order to perform this analysis, we assume the *spreading tail* (*ST*) agent model, presented in Marcolino et al. (2014) [57]. The *ST* agent model was developed to study how

teams of voting agents change in performance as the size of the action space increases. The basic assumption is that the pdf of each member of the team has a non-zero probability over an increasingly larger number of suboptimal actions as the action space grows, while the probability of voting for the optimal action remains unchanged. Marcolino et al. (2014) [57] perform an experimental validation of this model in the Computer Go domain.

For completeness, we briefly summarize here the formal definition of the *ST* agent model, and we refer the reader to Marcolino et al. (2014) [57] for a more detailed description. Let $\mathbf{D_m}$ be the set of suboptimal actions ($a_j, j \neq 0$) assigned with a nonzero probability in the pdf of an agent i, and $d_m = |\mathbf{D_m}|$. The *ST* model assumes that there is a bound in the ratio of the suboptimal action with highest probability and the one with lowest nonzero probability, i.e., let $p_{i,min} = min_{j \in \mathbf{D_m}} p_{i,j}$ and $p_{i,max} = max_{j \in \mathbf{D_m}} p_{i,j}$; there is a constant ζ such that $p_{i,max} \leq \zeta p_{i,min}$ $\forall$ agents i. *ST* agents are agents whose d_m is non-decreasing on m and $d_m \to \infty$ as $m \to \infty$. Marcolino et al. (2014) [57] consider that there is a constant $\varepsilon > 0$, such that for all *ST* agents i, $\forall m$, $p_{i,0} \geq \varepsilon$. They also assume that $p_{i,0}$ does not change with m.

Let the size of the action space $|\mathbf{A}| = \rho$, and $p_{i,j}$ be the probability that agent i votes for action with rank j. Marcolino et al. (2014) [57] show that when $\rho \to \infty$, the probability that a team of n *ST* agents will play the optimal action converges to:

$$\tilde{p}_{best} = 1 - \prod_{i=1}^{n}(1 - p_{i,0}) - \sum_{i=1}^{n}(p_{i,0} \prod_{j=1, j \neq i}^{n}(1 - p_{j,0})) \frac{n-1}{n}, \quad (6)$$

that is, the probability of two or more agents agreeing over suboptimal actions converges to zero, and the agents can only agree over the optimal choice (note that a suboptimal action can still be taken as we may have situations where no agent agrees). Hence, Equation 6 calculates the total probability minus the cases where the best action is not chosen: the second term covers the case where all agents vote for a suboptimal action and the third term covers the case where one agent votes for the optimal action and all other agents vote for suboptimal actions.

Before proceeding to our study, we are going to make a few definitions and then two weak assumptions. We consider now here any action space size. Let α be the probability of a team taking the optimal action when all agents disagree. Since we can only take the optimal action if one agent votes for that action, α is a function of the probability of each agent voting for the optimal action. That is, we may have voting profiles where all agents disagree and no agent voted for the optimal action, or we may have voting profiles where all agents disagree, but there is one agent that voted for the optimal action (and, hence, we may still take the optimal action due to random tie braking).

Let β be the probability of a team taking the optimal action when there is some agreement on the voting profile. β may be different according to each voting profile, but we assume that we always have that $\beta < 1$ if $\rho < \infty$, and $\beta = 1$ if $\rho \to \infty$, according to the *ST* agent model. That is, if two or more agents agree, there is always some probability $q > 0$ that they are agreeing over a suboptimal action, and $q \to 0$ as $\rho \to \infty$.

We will make the following weak assumptions: (i) If there is no agreement, the team is more likely to take a suboptimal action than an optimal action. I.e., $\alpha < 1 - \alpha$; (ii) If there is agreement, there is at least one voting profile where the team is more likely to take an optimal action than a suboptimal action. That is, there is at least one β such that $\beta > 1 - \beta$.

Assumption (i) is weak, since $\alpha < 1/n$ (as we break ties randomly and there may be cases where no agent votes for the optimal action). Clearly $1/n < 1 - 1/n$ for $n > 2$. Assumption (ii) is also weak, because if we are given a team that is always more likely to take suboptimal actions than an optimal action for *any* voting profile, then a trivial predictor that

always outputs "failure" would be optimal (and, hence, we would not need a prediction at all). Therefore, assumption (i) and (ii) are satisfied for all situations of interest. We present now our result:

Theorem 2 *Let* $\mathbf{T}$ *be a set of* ST *agents. The quality of our prediction about the performance of* $\mathbf{T}$ *is the highest as* $\rho \to \infty$.

Proof Let's fix the problem to predicting performance at one world state. Hence, as we consider a single decision, there is a single $\mathbf{H}^i$ such that $f_{\mathbf{H}^i} = 1$, and $f_{\mathbf{H}^j} = 0 \ \forall j \neq i$. In order to simplify the notation, we denote by $\mathscr{H}$ the subset $\mathbf{H}^i$ corresponding to $f_{\mathbf{H}^i} = 1$. We also consider the performance of the team as "success" on that fixed world state if they take the optimal action, and as "failure" otherwise.

Let a *voting event* be the process of querying the agents for the vote, obtaining the voting profile and the corresponding final decision. Hence, it has a unique correct label ("success" or "failure"). A voting event ξ will be mapped to a point χ in the feature space, according to the subset of agents that agreed on the chosen action. Multiple voting events, however, will be mapped to the same point χ (as exactly the same subset can agree in different situations, sometimes they may be agreeing over the optimal action, and sometimes they may be agreeing over suboptimal actions). Hence, given a point χ, there is a certain probability that the team was successful, and a certain probability that the team failed. Therefore, by assigning a label to that point, our predictor will also be correct with a certain probability. With enough data, the predictor will output the more likely of the two events. That is, if given a profile, the team has a probability p of taking the optimal action, and $p > 1-p$, the predictor will output "success", and it will be correct with probability p. Correspondingly, if $1-p > p$, the predictor will output "failure", and it will be correct with probability $1-p$. Hence, the probability of the prediction being correct will be $\max(p, 1-p)$.

We first study the probability of making a correct prediction across the whole feature space, for different action space sizes, and after that we will focus on what happens with the specific voting events as the action space changes.

Let us start by considering the case when $\rho \to \infty$. By Equation 6, we know that every time two or more agents agree on the same action, that action will be the optimal one. Note that this is a very clear division of the feature space, as for every single point where $|\mathscr{H}| \geq 2$ the team will be successful with probability 1. Therefore, on this subspace we can make perfect predictions.

The only points in the feature space where a team may still take a suboptimal action are the ones where a single agent agrees on the chosen action, i.e., $|\mathscr{H}| = 1$. Hence, for such points we will make a correct prediction with probability $\max(\alpha, 1-\alpha)$.

Let's now consider cases with a smaller action space size (i.e., $\rho < \infty$). Let's first consider the subspace $|\mathscr{H}| \geq 2$. Before, our predictor was correct with probability 1. Now, given a voting event where there is an agreement, there will be a probability $\beta < 1$ of the team taking the optimal action. Hence, the predictor will be correct with probability $\max(\beta, 1-\beta)$, but $\max(\beta, 1-\beta) < 1$.

Let's consider now the subspace $|\mathscr{H}| = 1$. Here the quality of the prediction depends on α, which is a function of the probability of each agent playing the best action. On the *ST* agent model, however, the probability of one agent voting for the best action is independent of ρ [57]. Hence, α does not depend on the action space size, and for these cases the quality of our prediction will be the same as before. Therefore, for all points in the feature space, the probability of making a correct prediction is either the same or worse when $\rho < \infty$ than when $\rho \to \infty$.

However, that does not complete the proof yet, because a voting event ξ may map to a different point χ when the action space changes. For instance, the number of agents that agree over a suboptimal action may overpass the number of agents that agree on the optimal action as the action spaces changes from $\rho \to \infty$ to $\rho < \infty$. Therefore, we need to show that our prediction will be strictly better when $\rho \to \infty$ irrespective of such mapping.

Hence, let us now study the voting events. As the number of actions decrease, a certain voting event ξ when $\rho \to \infty$, will map to a voting event ξ' when $\rho < \infty$ (where ξ may or may not be equal to ξ'). Let χ and χ' be the corresponding points in the feature space for ξ and ξ'. Also, let $\mathscr{H}$ and $\mathscr{H}'$ be the respective subset of agreeing agents. Let's consider now the four possible cases:

(i) $|\mathscr{H}| = |\mathscr{H}'| = 1$. For such events, the performance of the predictor will remain the same, that is, for both cases we will make a correct prediction with probability $\max(\alpha, 1-\alpha)$. Note that this case will not happen for all events, as $p_{i,0} \not\to 0$ when $\rho \to \infty$, hence there will be at least one event where $|\mathscr{H}| \geq 2$.

(ii) $|\mathscr{H}| \geq 2$, $|\mathscr{H}'| \geq 2$. For such events the performance of the predictor will be higher when $\rho \to \infty$, as we can make a correct prediction for a point χ with probability 1, while for a point χ' with probability $\max(\beta, 1-\beta) < 1$.

(iii) $|\mathscr{H}| \geq 2$, $|\mathscr{H}'| = 1$. This case will not happen under the *ST* agent model. If there was a certain subset $\mathbf{H}$ of agreeing agents when $\rho \to \infty$, when we decrease the number of actions the new subset of agreeing agents $\mathbf{H}'$ will either have the same size or will be larger. This follows from the fact that we may have a larger subset agreeing over some suboptimal action when the action space decreases, but the original subset that voted for the optimal action will not change.

(iv) $|\mathscr{H}| = 1$, $|\mathscr{H}'| \geq 2$. We know that in this case ξ' is an event where the team fails (otherwise the same subset would also have agreed when $\rho \to \infty$). Hence, for $1-\alpha > \alpha$ (weak assumption (i)), we make a correct prediction for such case when $\rho \to \infty$. When $\rho < \infty$, we make a correct prediction if $1-\beta > \beta$. β, however, depends on the voting profile of the event ξ'. By weak assumption (ii), there will be at least one event where the team is more likely to be correct than wrong (that is, $\beta > 1-\beta$). Hence, there will be at least one event where our predictor changes from making a correct prediction (when $\rho \to \infty$) to making an incorrect prediction (when $\rho < \infty$).

Hence, for all voting events, the probability of making a correct prediction will either be the same or worse when $\rho < \infty$ than when $\rho \to \infty$, and there will be at least one voting event where it will be worse, completing the proof. Hence, $\rho \to \infty$ is strictly the best case for our prediction. As we assume that all world states are independent, if $\rho \to \infty$ is the best case for a single world state, it will also be the best case for a set of world states. □

5 Results

5.1 Computer Go

We first test our prediction method in the Computer Go domain. We use four different Go software: Fuego 1.1 [26], Gnu Go 3.8 [27], Pachi 9.01 [9], MoGo 4 [30], and two (weaker) variants of Fuego (FuegoΔ and FuegoΘ), in a total of six different, publicly available, agents. Fuego is the strongest agent among all of them [56]. The description of FuegoΔ and FuegoΘ is available in Marcolino et al. (2014) [57].

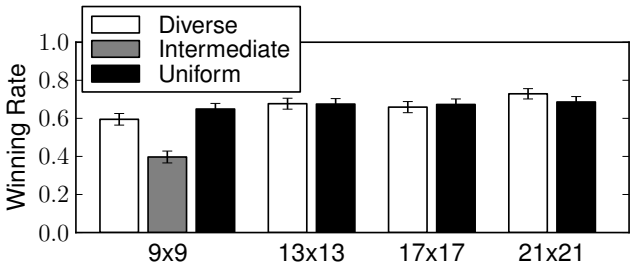


Fig. 3: Winning rates of the three teams, under four different board sizes.

We study three different teams: *Diverse*, composed of one copy of each agent[2]; *Uniform*, composed of six copies of the original Fuego (initialized with different random seeds, as in Soejima et al. (2010) [71]); *Intermediate*, composed of six random parametrized versions of Fuego (from Jiang et al. (2014) [38]). In all teams, the agents vote together, playing as white, in a series of Go games against the original Fuego playing as black. We study four different board sizes for *diverse* and *uniform*: 9x9, 13x13, 17x17 and 21x21. For *intermediate*, we study only 9x9, since the random parametrizations of Fuego do not work on larger boards. In Go a player is allowed to place a move at any empty intersection of the board, so the largest number of possible actions at each board size is, respectively: 81, 169, 289, 441. Computation for the work described in this paper was supported by the University of Southern California's Center for High-Performance Computing (`http://hpcc.usc.edu`). We ran the Go games in HP DL165 machines, each with 12 2.33 GHz cores, and 48GB of RAM.

In order to evaluate our predictions, we use a dataset of 1000 games for each team and board size combination (in a total of 9000 games, all played from the beginning). For all results, we used repeated random sub-sampling validation. We randomly assign 20% of the games for the testing set (and the rest for the training set), keeping approximately the same ratio as the original distribution. The whole process is repeated 100 times. Hence, in all graphs we show the average results, and the error bars show the 99% confidence interval ($p = 0.01$), according to a *t-test*. If the error bars cannot be seen in a certain point in a graph, it is because they are smaller than the symbol used to mark that point in the graph. Moreover, when we say that a certain result is significantly better than another, we mean statistically significantly better, according to a *t-test* where $p < 0.01$, unless we explicitly give a p value.

First, we show the winning rates of the teams in Figure 3. This result is not yet evaluating the quality of our prediction, it is merely background information that we will use when analyzing our prediction results later. On 9x9 Go, *uniform* is better than *diverse* with statistical significance ($p = 0.014$), and both teams are clearly significantly better than *intermediate* ($p < 2.2 \times 10^{-16}$). On 13x13 and 17x17 Go, the difference between *diverse* and *uniform* is not statistically significant ($p = 0.9619$ and 0.5377, respectively). On 21x21 Go, the *diverse* team is significantly better than *uniform* ($p = 0.03897$).

[2] Except for Gnu Go, all other agents use Monte Carlo Tree Search [12]. However, each agent was developed by different groups, and use different heuristics and implementation strategies. In comparison with the other teams studied, this is the team with the greater diversity. An empirical evaluation of the diversity of this team is available at Marcolino et al. (2014) [58].

In order to verify our online predictions, we used the evaluation of the original Fuego, but we give it a time limit 50× longer (i.e., it runs for 500*s* per evaluation)[3]. We will refer to this version as "Baseline". We, then, use the Baseline's evaluation of a given board state to estimate its probability of victory, allowing a comparison with our approach. Considering that an evaluation above 0.5 is "success" and below is "failure", we compare our predictions with the ones given by the Baseline's evaluation, at each turn of the games.

We use this method because the likelihood of victory changes dynamically during a game. That is, a team could be in a winning position at a certain stage, after making several good moves, but suddenly change to a losing position after committing a mistake. Similarly, a team could be in a losing position after several good moves from the opponent, but suddenly change to a winning position after the opponent makes a mistake. Therefore, simply comparing with the final outcome of the game would not be a good evaluation. However, for the interested reader, we show in the appendix how the evaluation would be comparing with the final outcome of the game — and we note here that our prediction quality is still high under that alternative evaluation method.

Since the games have different lengths, we divide all games in 20 stages, and show the average evaluation of each stage, in order to be able to compare the evaluation across all games uniformly. Therefore, a stage is defined as a small set of turns (on average, 1.35 ± 0.32 turns in 9×9; 2.76 ± 0.53 in 13×13; 4.70 ± 0.79 in 17×17; 7.85 ± 0.87 in 21×21). For all games, we also skip the first 4 moves, since our baseline returns corrupted information in the beginning of the games.

We will present our results in five different sections. First, we will show the analysis for a fixed threshold (that is, the value ϑ above which the output of our prediction function $\hat{f}$ will be considered "success"). Secondly, we are going to analyze across different thresholds using receiver operating characteristic (ROC) curves, and the area under such curves (AUC). Thirdly, we will present the results for different board sizes. Then, finally, we will study the performance of the reduced feature vector, and we will show results when playing against a different adversary.

5.1.1 Single Threshold Analysis

We start by showing our results for a fixed threshold, since it gives a more intuitive understanding of the quality of our prediction technique. Hence, in these results we will consider that when our prediction function $\hat{f}$ returns a value above 0.5 for a certain game, it will be classified as "success", and when the value is below 0.5, it will be classified as "failure". In this section we also restrict ourselves to the 9x9 Go case.

We will evaluate our prediction results according to five different metrics: Accuracy, Failure Precision, Success Precision, Failure Recall, Success Recall. Accuracy is defined as the sum of the true positives and true negatives, divided by the total number of tests (i.e., true positives, true negatives, false positives, false negatives). Hence, it gives an overall view of the quality of the classification. Precision gives the percentage of data points classified

[3] As this evaluation is much more computationally expensive than game playing, we allowed it to run across different computer types in the HPC cluster, ranging from 2.0 GHz to 3.0 GHz. The full list of hardware is available at `https://hpcc.usc.edu/support/infrastructure/hpcc-computing-resource-overview/`. Note that keeping the hardware fixed here is not as critical as in game playing – as the hardware could affect the outcome of a game, but here we only want to obtain an estimation of the value of a position. Additionally, our conclusions still hold when we compare our predictions with the actual final outcome of the games (see appendix).

with a certain label ("success" or "failure") that are correctly labeled. Recall denotes the percentage of data points that truly pertain to a certain label, that are correctly classified.

We show the results in Figure 4. As we can see, we were able to obtain a high-accuracy very quickly, already crossing the 0.5 line in the 2^{nd} stage for all teams. In fact, the accuracy is significantly higher than the 0.5 mark for all teams after the 2^{nd} stage (and for *diverse* and *intermediate* since the 1^{st} stage).

From around the middle of the games (stage 10), the accuracy for *diverse* and *uniform* already gets close to 60% (with *intermediate* only close behind). Although we can see some small drops – that could be explained by the sudden changes in the game –, overall the accuracy increases with the game stage number, as expected. Moreover, for most of the stages, the accuracy is higher for *diverse* than for *uniform*. The prediction for *diverse* is significantly better than for *uniform* in 90% of the stages. It is also interesting to note that the prediction for *intermediate* is significantly better than for *uniform* in 60% of the stages, even though *intermediate* is a significantly weaker team. In fact, we can see that in the last stage, the accuracy, the failure precision and the failure recall is significantly better for *intermediate* than for the other teams.

5.1.2 Multiple Threshold Analysis

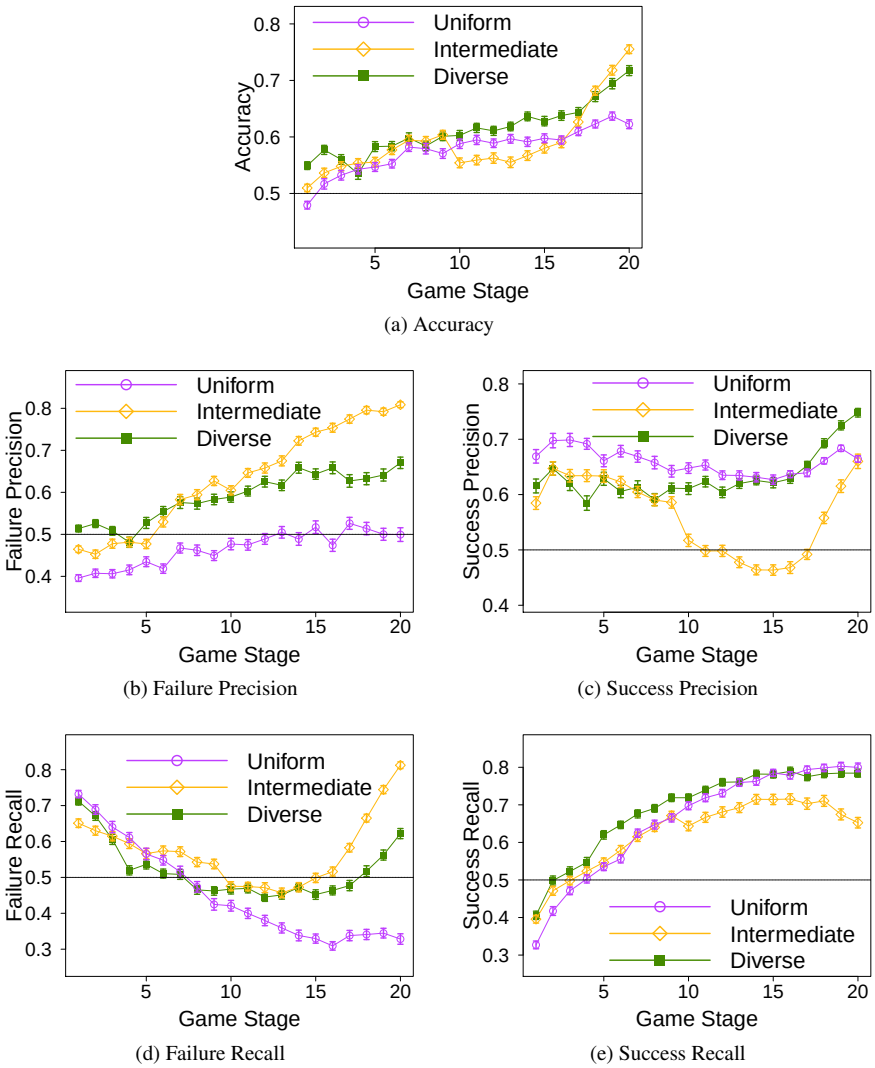
We now measure our results under a variety of thresholds (that is, the value ϑ above which the output of our prediction function $\hat{f}$ will be considered "success"). In order to perform this study, we use receiver operating characteristic (ROC) curves. An ROC curve shows the true positive and the false positive rates of a binary classifier at different thresholds. The true positive rate is the number of true positives divided by the sum of true positives and false negatives (i.e., the total number of items of a given label). That is, the true positive rate shows the percentage of "success" cases that are correctly labeled as such. The false positive rate, on the other hand, is the number of false positives divided by the sum of false positives and true negatives (i.e., the total number of items that *are not* of a given label). That is, the false positive rate shows the percentage of "failure" cases that are wrongly classified as "success".

Hence, ROC curves allow us to better understand the trade-off between the true positives and the false positives of a given classifier. By varying the threshold, we can increase the number of items that receive a given label, increasing henceforth the number of items of such label that are correctly classified, but at the cost of also increasing the number of items that incorrectly receive the same label.

The ideal classifier is the one on the upper left corner of the ROC graph. This point (true positive rate = 1, false positive rate = 0) indicates that every item classified with a certain label truly pertains to such label, and every item that truly pertains to such label receives the correct classification.

In this paper, we do not aim only on studying the performance of a single classifier, but rather on comparing the performance of our prediction technique on a variety of situations, changing the team and the action space size (i.e., the board size). Hence, we also study the area under the ROC curve (AUC), as a way to synthesize the quality information from the curve into a single number. The higher the AUC, the better the prediction quality of our technique in a given situation. That is, a completely random classifier would have an AUC of 0.5, and as the ROC curve moves towards the top-left corner of the graph, the AUC gets closer and closer to 1.0.

In fact, the AUC metric has been shown to be equal to the probability of, given a pair of items with different labels (one "success" case and one "failure" case, in our situation),



(a) Accuracy

(b) Failure Precision

(c) Success Precision

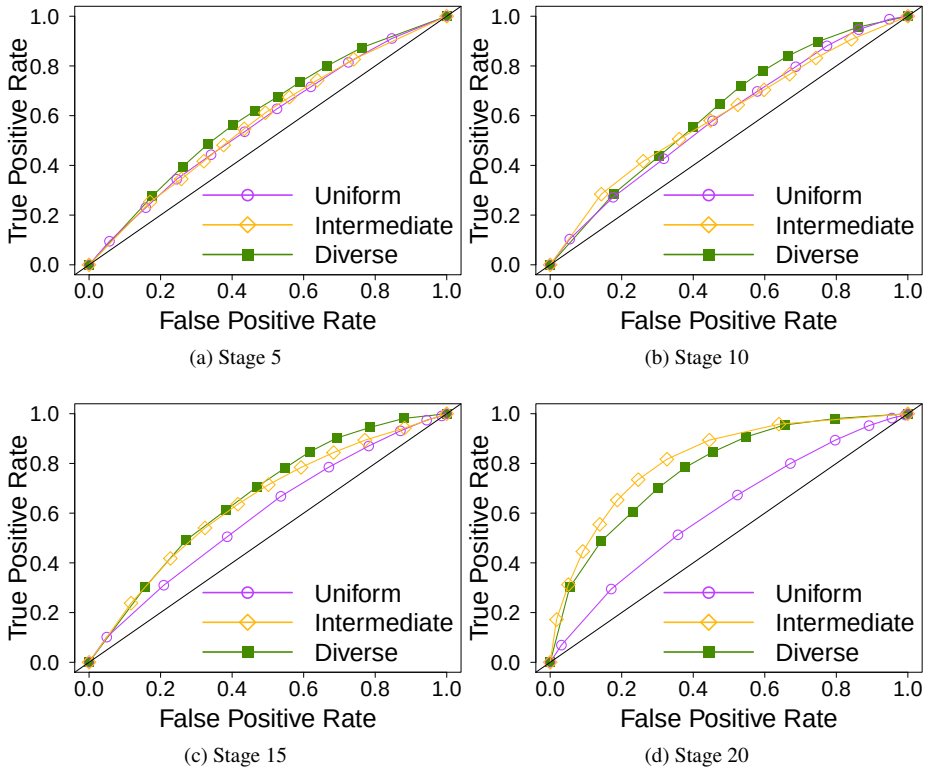
(d) Failure Recall

(e) Success Recall

Fig. 4: Performance metrics over all turns of 9x9 Go games.

correctly considering the item that was truly a "success" as more likely to be a "success" case than the other item. It has also been shown to be related to other important statistical metrics, such as the *Wilcoxon test of ranks*, the *Mann-Whitney U* and the *Gini coefficient* [33,59,32].

We start by studying multiple thresholds in a fixed board size (9x9), and in the next section we study the effect of increasing the action space. Hence, in Figure 5 we show the ROC curves for all teams in the 9x9 Go games, for 4 different stages of the game. Although in the beginning of the game (stage 5) it is hard to distinguish the results for the different teams, we can note that from the middle of the games to the end (stage 15 and 20),



(a) Stage 5 (b) Stage 10

(c) Stage 15 (d) Stage 20

Fig. 5: ROC curves, analyzing different thresholds in 9x9 Go.

there is a clear distinction between the prediction quality for *diverse* and *intermediate*, when compared with the one for *uniform*, across many different thresholds.

As mentioned, to formally compare these results across all stages we use the AUC metric. Hence, in Figure 6, we show the AUC for the three different teams in 9x9 Go. As we can see, the three teams have similar AUCs up to stage 10, but from that stage on we can get better AUCs for both *diverse* and *intermediate*, significantly outperforming the AUC for *uniform* in all stages. We also find that, considering all stages, we have a significantly better AUC for *diverse* (than for *uniform*) in 85% of the cases, and for *intermediate* in 55% of the cases. Curiously, we can also note that even though *intermediate* is the weakest team, we can obtain for it the (significantly) best AUC in the last stage of the games, surpassing the AUC found for the other teams. Overall, these results show that our hypothesis that we can make better predictions for teams that have higher diversity holds irrespective of the threshold used for classification. On the next section, we study the effect of increasing the action space size.

5.1.3 Action Space Size

Let us start by showing, in Figure 7, ROC curves for *diverse* and *uniform* under different board sizes. It is harder to distinguish the curves on stages 5 and 10, but we can notice that

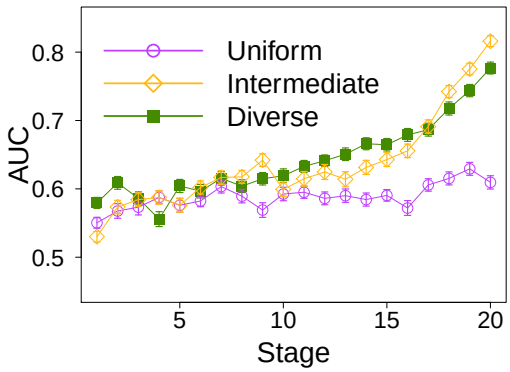


Fig. 6: AUC for *diverse*, *uniform* and *intermediate* teams, in 9x9 Go.

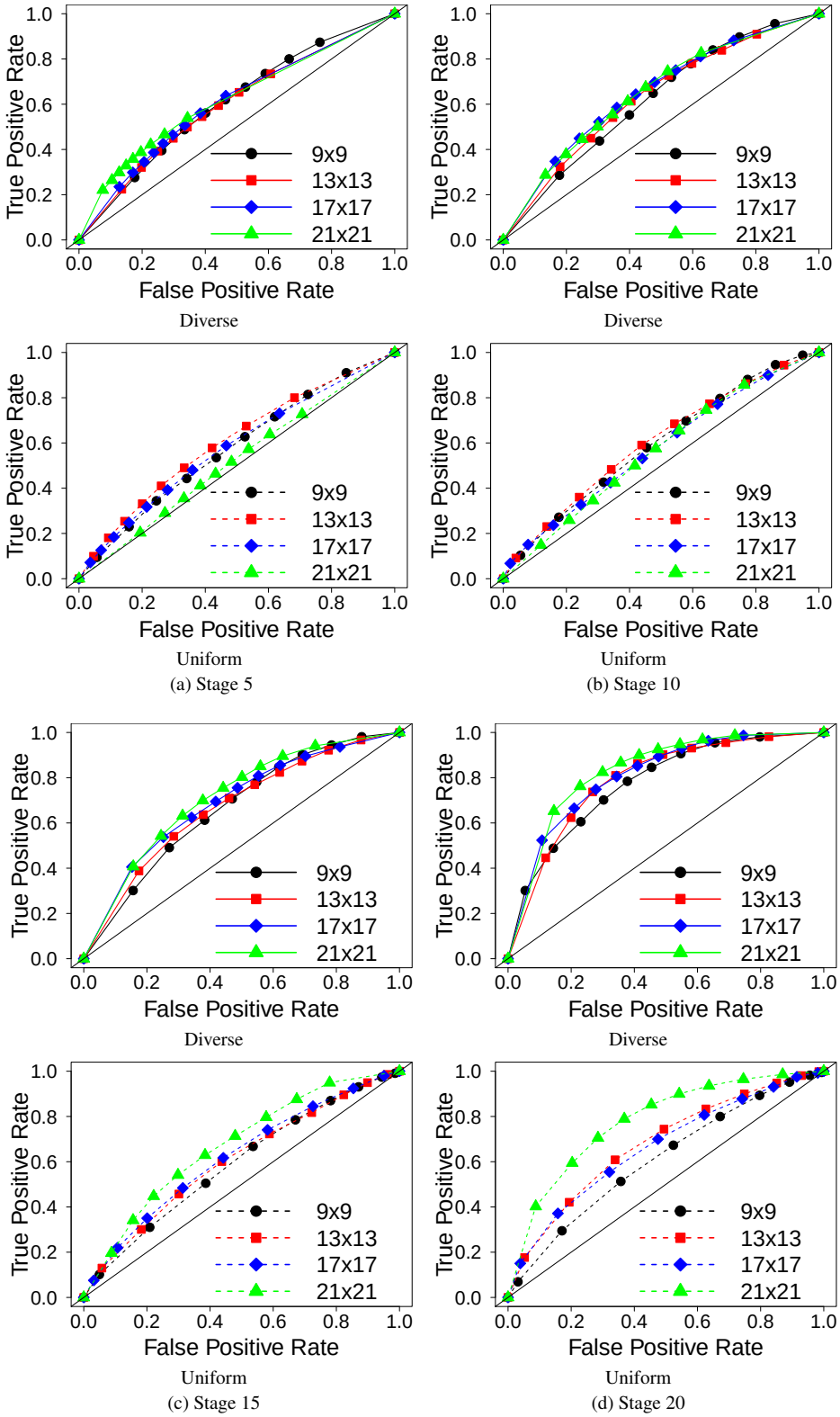
the curve for 21x21 tends to dominate the others on stages 15 and 20. Again, to better study these results, we look at how the AUC changes for different teams and board sizes.

In Figure 8 we can see the AUC results. For the *diverse* team (Figure 8 (a)), we start observing the effect of increasing the action space after stage 5, when the curves for 17×17 and 21×21 tends to dominate the other curves. In fact, the AUC for 17×17 is significantly better than smaller boards in 60% of the stages, and in 80% of the stages after stage 5. Moreover, after stage 5 no smaller board is significantly better than 17×17. Concerning 21×21, we can see that from stage 14, its curve completely dominates all the other curves. In all stages from 14 to 20 the result for 21×21 is significantly better than for all other smaller boards. Hence, we can note that the effect of increasing the action space seems to depend on the stage of the game, and it gets more evident as the stage number increases.

Concerning the *uniform* team (Figure 8 (b)), up to 17×17 we cannot observe a positive impact of the action space size on the prediction quality; but for 21×21 there is clearly an improvement from the middle game when compared with smaller boards. On all 8 stages from stage 13 to stage 20, the result for 21×21 is significantly better than for other board sizes.

In terms of percentage of stages where the result for 21×21 is significantly better than for 9×9, we find that it is 40% for the *uniform* team, while it is 85% for the *diverse* team. Hence, the impact of increasing the action space occurs for *diverse* earlier in the game, and over a larger number of stages.

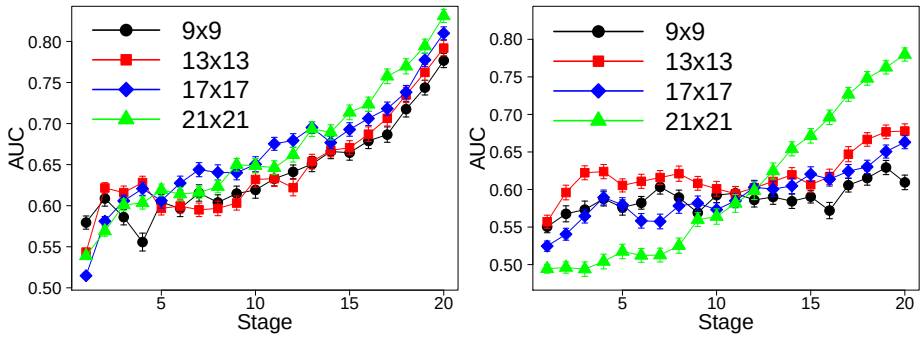
Now, in order to compare the performance for *diverse* and *uniform* under different board sizes, we show the AUCs in Figure 9 organized by the size of the board (Figure 6 is repeated here in Figure 9 (a) to make it easier to observe the difference between board sizes). It is interesting to observe that the quality of the predictions for *diverse* is better than for *uniform*, irrespective of the size of the action space. Moreover, while for 9×9 and 13×13 the prediction for *diverse* is only always significantly better than for *uniform* after around stage 10, we can notice that for 17×17 and 21×21, the prediction for *diverse* is always significantly better than for *uniform*, irrespective of the stage (except for stage 1 in 17×17). In fact, we can also show that the difference between the teams is greater on larger boards. In Figure 10 we can see the difference between *diverse* and *uniform*, in terms of area under the AUC graph, and also in terms of percentage of stages where *diverse* is significantly better than *uniform*, for 9×9 and 21×21. The difference between the areas in 9×9 and 21×21 is statistically significant, with $p = 0.0003337$.



(a) Stage 5 (b) Stage 10

(c) Stage 15 (d) Stage 20

Fig. 7: ROC curves for *diverse* and *uniform*, for different board sizes.



(a) Diverse Team (b) Uniform Team

Fig. 8: AUC for different teams and board sizes, organized by teams.

We study again the accuracy, precision and recall in 21×21, as the results in these metrics may be more intuitive to understand for most readers (although limited to a fixed threshold, in this case 0.5). We show these results in Figure 11, where for comparison we also plot the results for 9×9. For the *diverse* team, the accuracy in 21×21 is consistently significantly better than in 9×9, in all stages from stage 12. We also have a significantly better accuracy in the first 5 stages. Concerning *uniform*, the accuracy for 21×21 is only consistently better from stage 15 (and also in stage 1). Hence, again, we can notice a better prediction quality on larger boards, and also a higher impact of increasing the action space size on *diverse* than on *uniform* (in terms of number of stages, and also how early the performance improves).

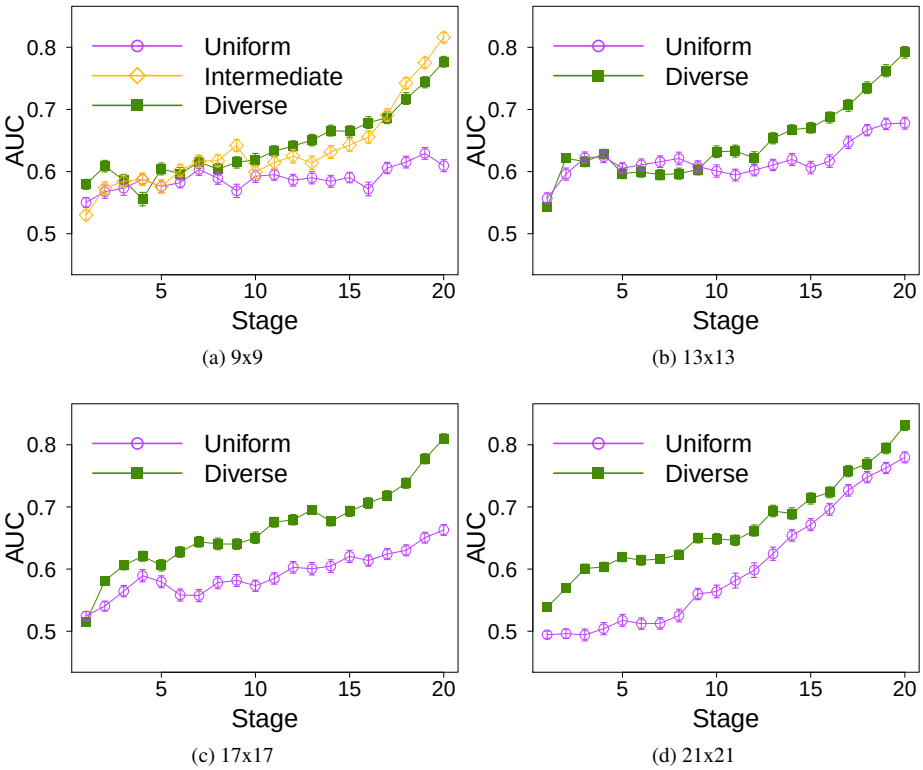
5.1.4 Reduced Feature Vector

Finally, we study the reduced feature vector. The results are very similar to the ones using the full feature vector, so we do not repeat them here due to space constraints. We note, however, that we can obtain similar results with a much more scalable representation[4]. In fact, in Figure 12 we study the area under the AUC graphs of the full and reduced representations, for all combinations of teams and board sizes. Surprisingly, the reduced representation turns out to be significantly better for all teams on the 9×9 and 13×13 boards. This may happen since it is easier to learn a model with less features. For the *diverse* team, however, the importance of the full representation increases as the action space grows. On 17×17, the difference between the representations is not statistically significant ($p = 0.9156$), while on 21×21 the full representation is actually significantly better than the reduced one ($p = 0.001794$).

5.1.5 Different Adversaries

We also tested the performance of our technique when Go games are played against a different adversary, in order to show that our approach still works under different situations.

[4] All our main conclusions still hold, except that this time the AUC for 21×21 is significantly better than for 9×9 around stage 15 for both teams.



(a) 9x9 (b) 13x13

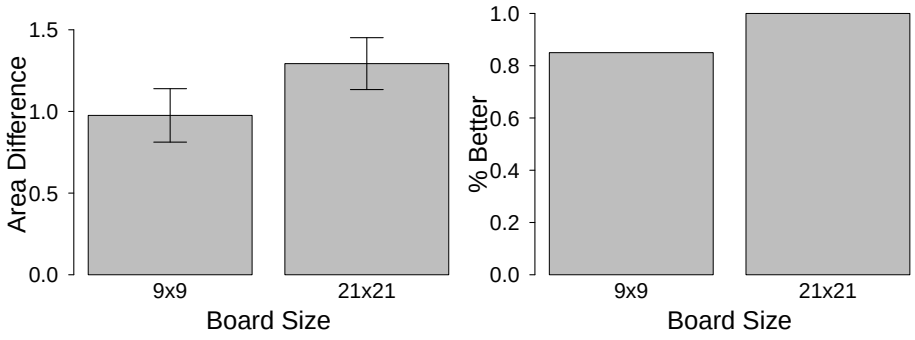
(c) 17x17 (d) 21x21

Fig. 9: AUC for different teams and board sizes, organized by board sizes.

Moreover, since the execution may not always match the training conditions, we also test our performance when playing against one adversary, but testing against another.

We will start by showing our results when testing and training against the same adversary. We will use the same settings as in the previous sections, except that instead of using a database of 1000 games against Fuego for each team, we are going to use a database of 1000 games for each team against Pachi. In this section we will focus on 9×9 Go games. For completeness, we first show the winning rates of the teams against Pachi in Figure 13. Again, we are not evaluating here the quality of our predictions, but giving background information for better understanding our results. In this case, both *diverse* and *uniform* are significantly better than *intermediate* ($p < 8.759 \times 10^{-8}$). However, the difference between *diverse* and *uniform* is not significant ($p = 0.4111$).

We will again verify our predictions by comparing against Fuego's evaluation, but with a time limit $50\times$ longer ("Baseline"). In the appendix we show the evaluation when comparing with the actual final outcome of the games. We start by a single threshold analysis (0.5). We show the results in Figure 14, when training and testing with games against Pachi. As we can see, our proposed methodology still works in games played against a different adversary. The accuracy is around 60% in the middle of the games, going all the way to around 70% for *diverse*, and 90% for *intermediate*. For *uniform*, the accuracy grows to around 65% until stage 19, but drops to 57% in the last stage.



(a) By area (b) By percentage of stages

Fig. 10: Differences in prediction quality for the *diverse* and *uniform* teams.

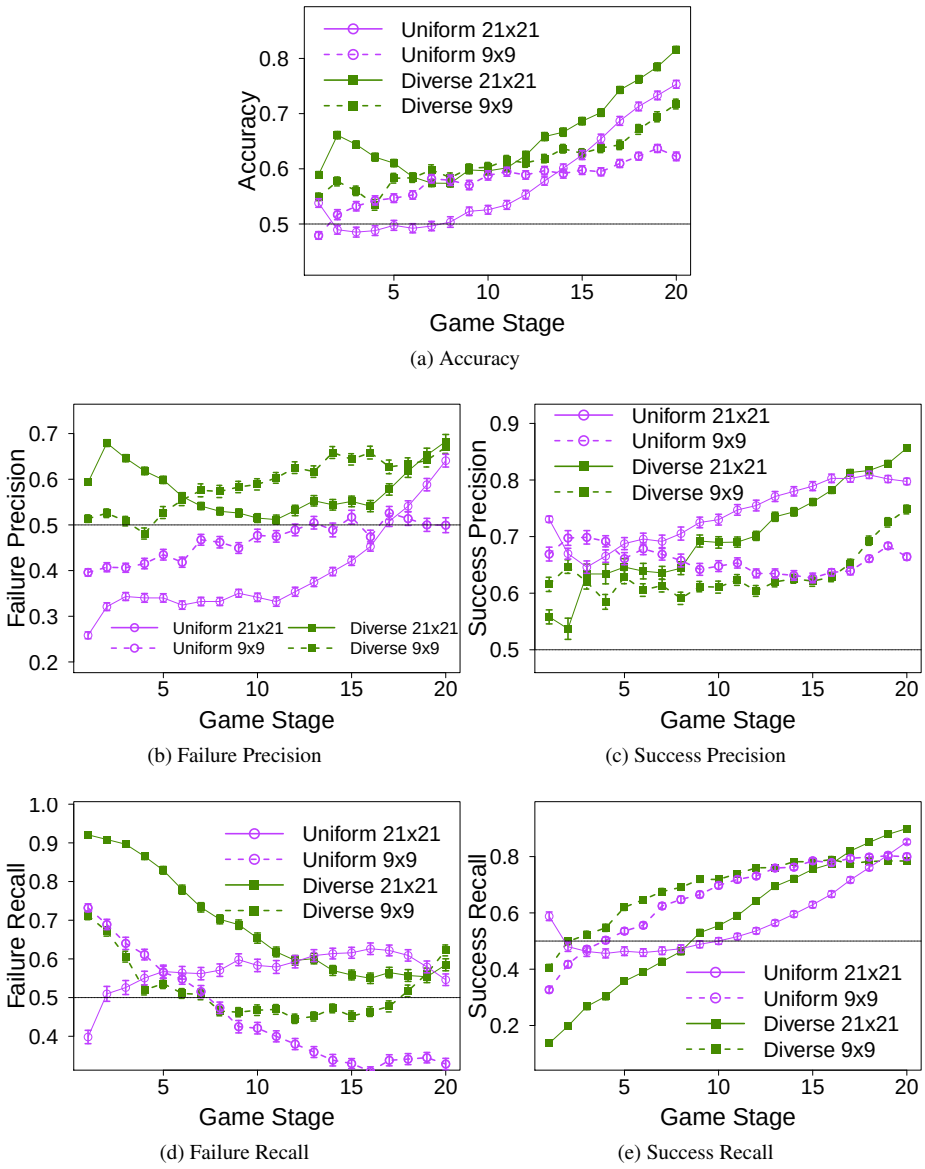
We now study our performance when training with games played against one adversary, but testing against another. For these results, we use the previous database (games against Fuego) for training, but we test our results in the database of games played against Pachi. We show the results in Figure 15. For easy comparison, we show by the dashed lines the previous result (training and testing against Pachi), and by the continuous line the current result (training against Fuego, and testing against Pachi).

As we can see, our technique still works well when training and testing against different adversaries. The curves have similar shapes to the ones of a fixed adversary, although in some stages the difference in performance is significant. Surprisingly, it is not necessarily the case that the performance is always worse when training and testing against different adversaries, as in some stages we actually find a significant better performance in that situation.

In terms of accuracy, the performance is better for the *diverse* team when training and testing against Pachi in all stages, except for the first and the last one, when the difference is not significant. For *uniform*, in the other hand, the performance is actually *better* when training against Fuego (and testing against Pachi) in 55% of the stages. Only in the last stage training against Pachi is significantly better, while in the other stages the difference is not statistically significant. Concerning the *intermediate* team, training against Fuego is *better* in 75% of the stages, and only in 10% of the stages it was better to train against Pachi. In the other stages the difference was not significant.

Similarly as before, we plot the ROC and AUC results, showing that our conclusions do not depend on a specific threshold. In Figure 16 we show the ROC curves, and in Figure 17 we show the respective AUC curves. As before, we show by the dashed line the result when training and testing against Pachi, and by the continuous line the result when training against Fuego, but testing against Pachi.

Hence, as we can see, our method does not depend on the adversary. Not only were we able to obtain high quality predictions when training and testing against Pachi (instead of Fuego), but also when training against Fuego, but testing against Pachi. This conclusion also holds when using the reduced feature vector. In Figure 18 (a) we show the area under the AUC graph for the full and reduced representation when training and testing against Pachi, while in Figure 18 (b) we show the area when training against Fuego, but testing against



(a) Accuracy

(b) Failure Precision

(c) Success Precision

(d) Failure Recall

(e) Success Recall

Fig. 11: Performance metrics over all turns of 9x9 and 21x21 Go games.

Pachi. The performance of the reduced representation is, again, close to the one of the full representation (and even statistically significantly better in most cases).

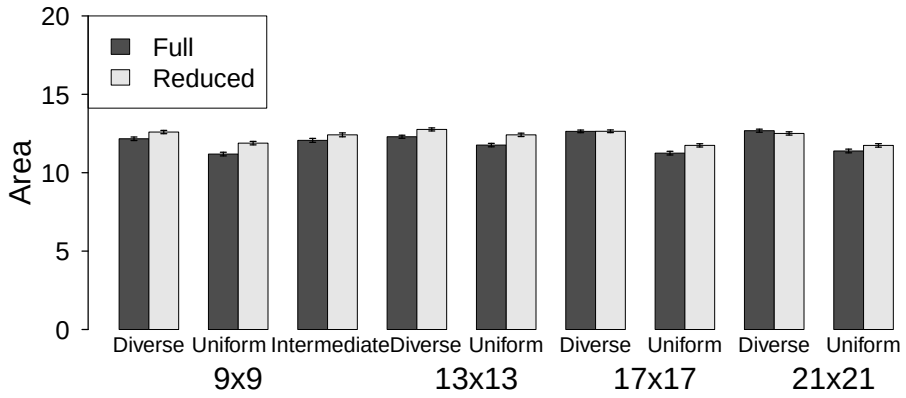


Fig. 12: Comparison of prediction quality with the full and reduced representation.

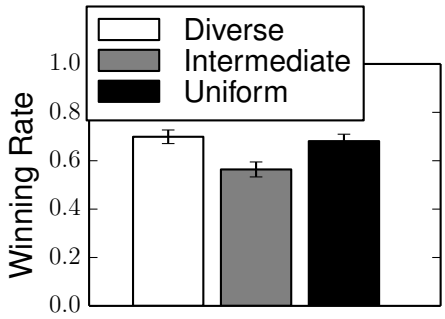
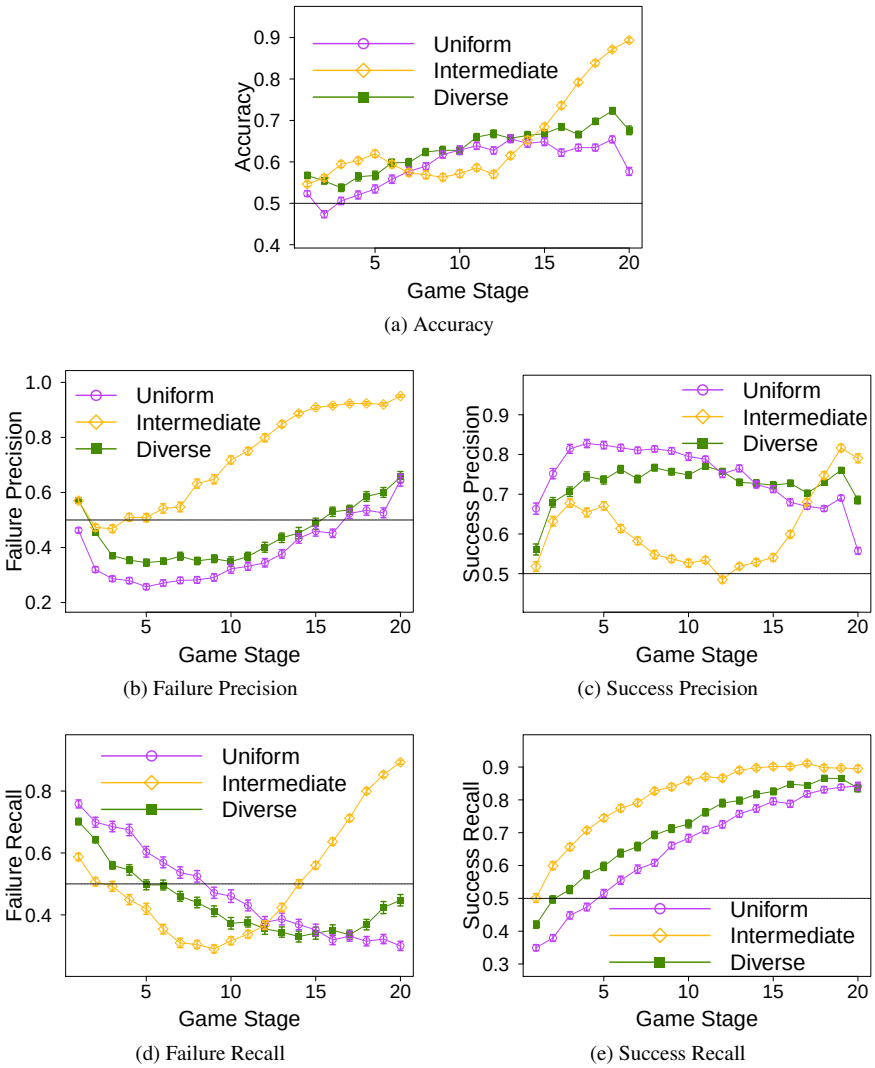


Fig. 13: Winning rates of the three teams, when playing against Pachi.

5.2 Ensemble Learning

In this section, we demonstrate that our approach also applies to other domains, by using our technique in order to predict the performance of an ensemble system. Note that we are not using here an ensemble system to predict the performance of a team. We are still using a single predictor function, but now the team of agents whose final performance we want to predict is a set of classifiers voting in order to assign labels to sets of items. Hence, an agent here corresponds to one classifier, an action corresponds to a label assignment, and a world state corresponds to an item.

We use the scikit-learn's digits dataset [65], which is composed of 1797 8×8 images. Each image is a hand-written digit, and the objective of the ensemble is to correctly identify the digit. Hence, for each item (i.e., image), there are 10 possible labels, corresponding to each possible digit, and only one of these labels will be a correct classification. We select 1000 items for training the agents. Among the chosen 1000 samples, 400 items are used in training each individual agent. We use the remaining 797 items for the testing phase, where the agents will vote to assign labels to sets of 50 items randomly sampled from the remaining 797. To avoid confusion, note that we are referring here to the training process of



(a) Accuracy

(b) Failure Precision

(c) Success Precision

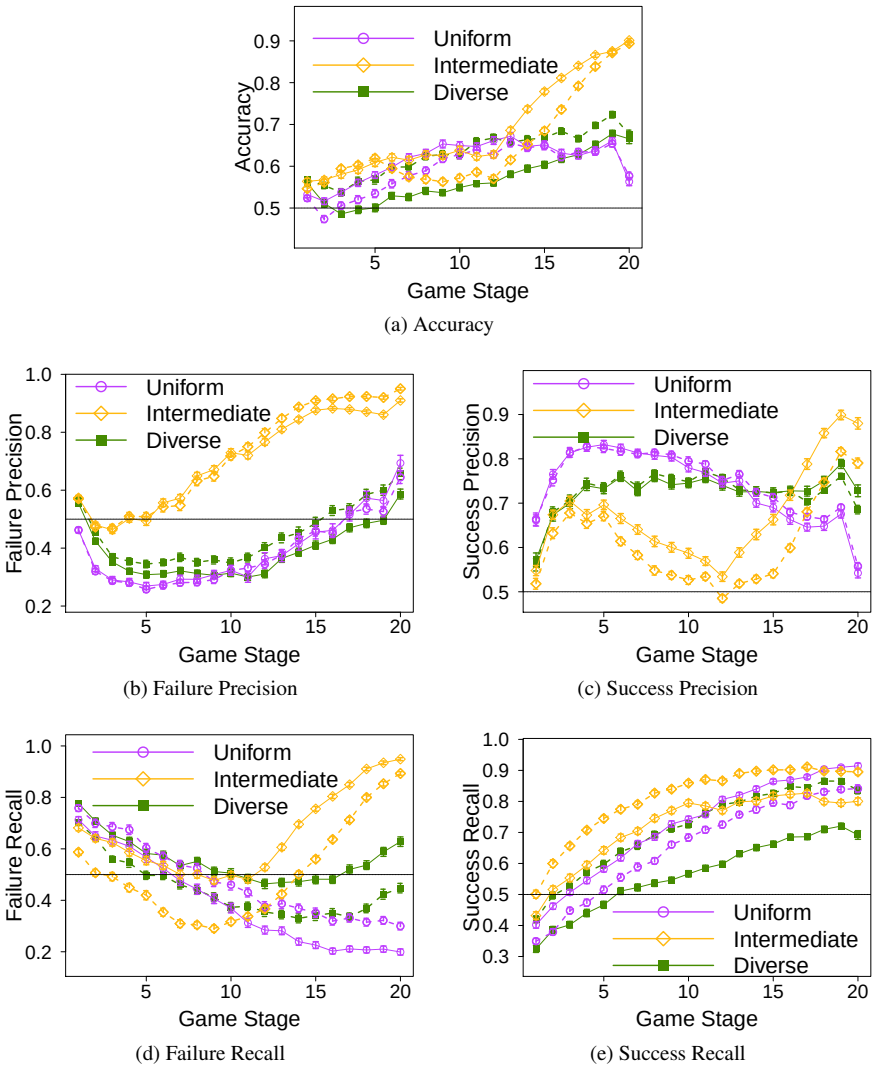
(d) Failure Recall

(e) Success Recall

Fig. 14: Performance metrics over all turns of 9x9 Go games, when playing against Pachi.

the individual agents, not of our team prediction methodology (which we will discuss later in this section).

We use 6 classification algorithms: Support Vector Machine (SVM) [20], Logistic Regression (LR) [22], Decision Tree (DT) [43], Nearest Neighbor (NN) [21], Bernoulli Naive Bayes (BNB) [54], Gaussian Naive Bayes (GNB) [54]. Each classification algorithm corresponds to one agent. We study two different teams. *Diverse 1*, composed of copies of the SVM agent with different parametrizations; and *Diverse 2*, composed of one copy of each agent. In Table 5 we show the parametrizations used in *Diverse 1*, according to the notation of the scikit-learn's library [65]. The parameters were selected in order for the individual agents to have different accuracies over the dataset, as a sign of diversity. In Figure 19 we



(a) Accuracy

(b) Failure Precision

(c) Success Precision

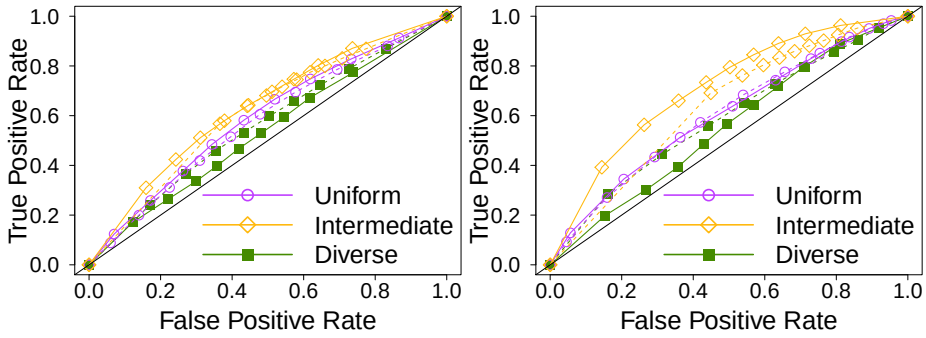
(d) Failure Recall

(e) Success Recall

Fig. 15: Performance metrics over all turns of 9x9 Go games, when training against Pachi (dashed line) or Fuego (continuous line), and testing against Pachi.

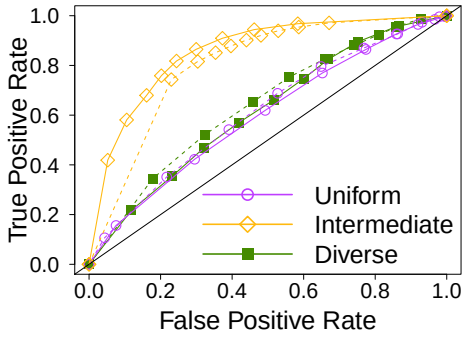
show the accuracy of each individual agents and the teams. Again, note that here we are not showing the accuracy of our prediction methodology, but reporting the performance of the agents and the teams as background information.

Unlike in the Go domain, where multiple copies of the same agent (like Fuego), would still vote for different actions due to the randomness involved in the rollouts of Monte Carlo Tree Search, a classifier is deterministic given its algorithm and parametrization. Therefore, we do not present results for *Uniform* in this domain, and instead we present two possible variations for a diverse team.

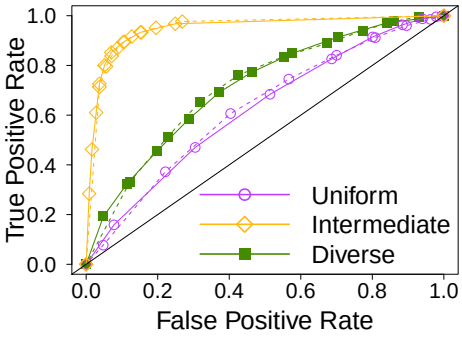


(a) Stage 5

(b) Stage 10



(c) Stage 15



(d) Stage 20

Fig. 16: ROC curves, analyzing different thresholds in 9x9 Go, when training against Pachi (dashed line) or Fuego (continuous line), and testing against Pachi.

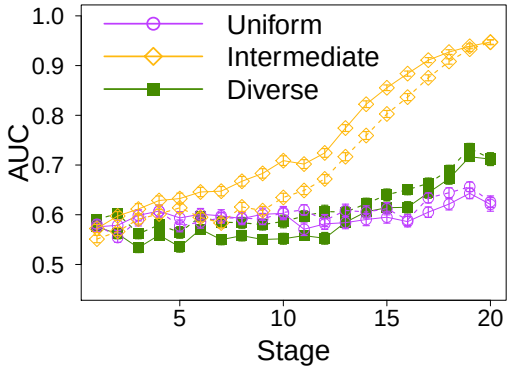
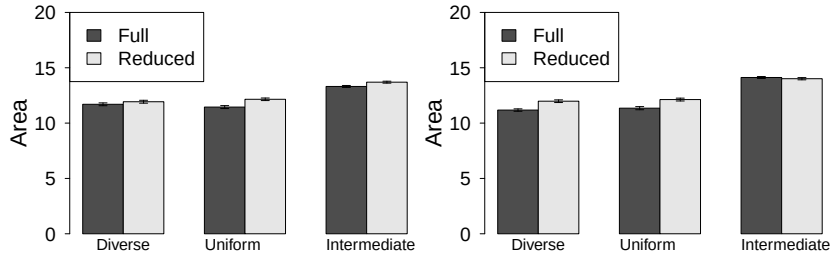


Fig. 17: AUC for *diverse*, *uniform* and *intermediate* teams, in 9x9 Go, when training against Pachi (dashed line) or Fuego (continuous line), and testing against Pachi.

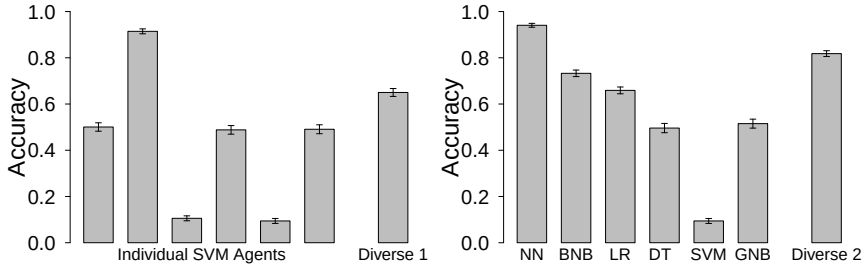


(a) Training Against Pachi (b) Training Against Fuego

Fig. 18: Comparison of prediction quality with the full and reduced representation, when testing against Pachi.

Agent	γ	C
Agent 1	0.01	1000
Agent 2	0.001	100
Agent 3	0.1	10000
Agent 4	0.01	10
Agent 5	0.1	1000
Agent 6	1×10^{-6}	10

Table 5: Different parametrizations of the SVM algorithm, used for building the *Diverse 1* team.



(a) Diverse 1 (b) Diverse 2

Fig. 19: Performance of the individual agents, and the ensemble system, in terms of accuracy.

We define one instance of the problem solving process as a set of 50 items to be classified. Similar to the previous domain, where we have a large set of games (each with a set of turns), we will have here multiple sets of 50 items to be classified (each corresponding to one world state). Each set of items is constructed from random sampling 50 items from our testing set (i.e., the 797 items that *were not* used to train the agents). We sample from two disjoint sets: one set of 400 items will be used when sampling 50 items for training our team prediction methodology. The remaining set of 397 items will be used when sampling 50 items for testing our team prediction methodology.

The agents will vote to assign labels for each item, and the problem solving process will be completed when the team assigns a label to each one of the 50 items. If the team classifies correctly more than a threshold percentage of the items, we consider it a "Success". Otherwise, the performance of the team is classified as "Failure". We explain below how the threshold is calculated.

We randomly generate 2500 sets in this fashion. We will now discuss the training of our team prediction methodology. We use 2000 sets for training, where the agents voted to classify each item, as discussed (note that the training set previously discussed in this section was used to train the individual agents). We use the mean performance over these 2000 sets as our threshold. Hence, if the team is able to classify a higher number of items than its average performance, we consider a "success". Otherwise, the problem solving process will be classified as "failure".

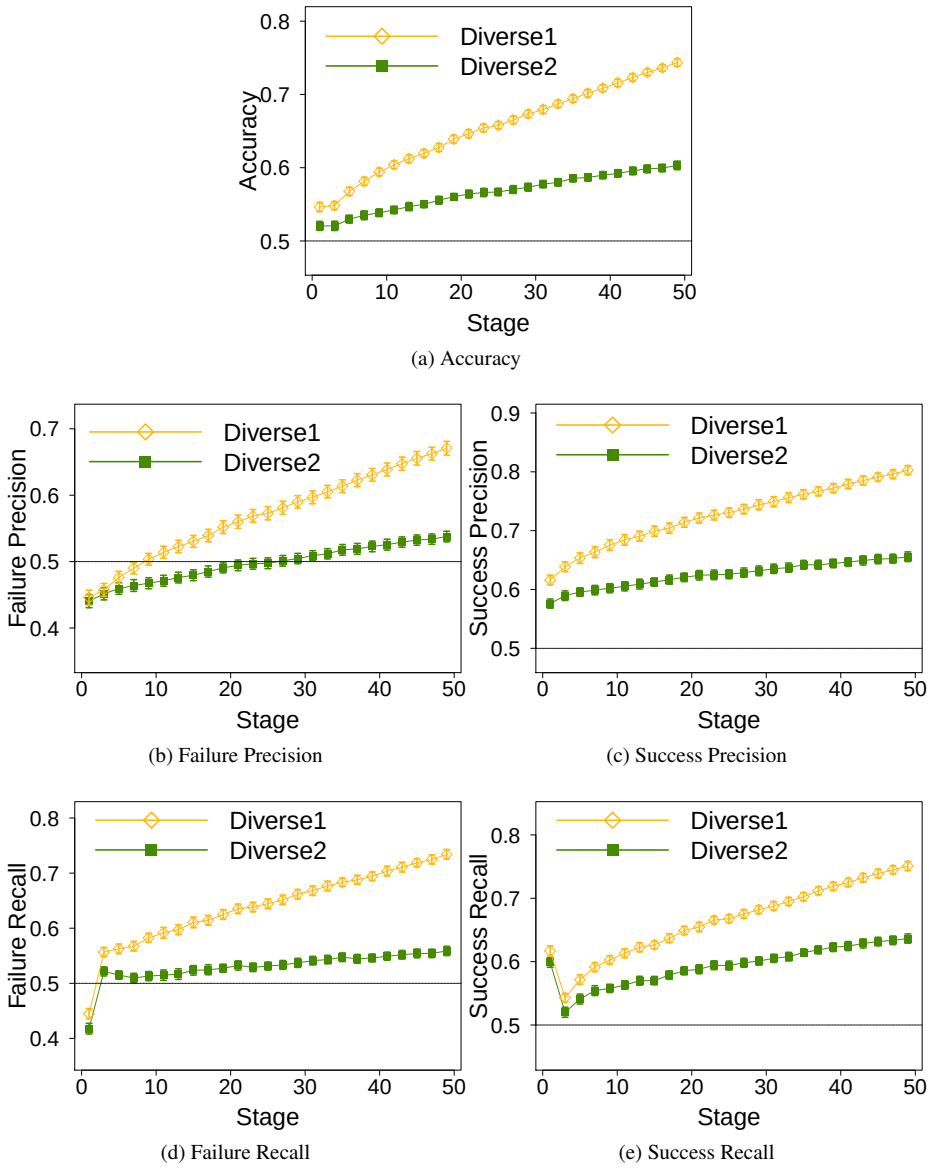
The remaining sets are used to test our team prediction methodology. We again evaluate accuracy, precision and recall. One run consists of the ensemble evaluating one set of items, and the performance metric at each stage will be calculated as the average over 500 runs (similarly as in the Computer Go case, where we evaluated across multiple games). The whole testing procedure is repeated 100 times; in all graphs we show the average results, and the error bars show the 99% confidence interval ($p = 0.01$), according to a *t-test*. When we say that a certain result is significantly better than another, we mean statistically significantly better, according to a *t-test* where $p < 0.01$, unless we explicitly give a p value.

We show the results in Figure 20, where each stage corresponds to an item to be classified. As we can see, we are able to obtain high quality predictions for both teams quickly. The accuracy for both teams is always significantly higher than 50%. Around the middle of the runs (i.e., Stage 25), the accuracy of *Diverse 1* is already around 65%, and *Diverse 2* 56%. Towards the end of the runs we obtain around 74% accuracy for *Diverse 1* and 60% accuracy for *Diverse 2*. In Figure 21, we can see the results using the reduced feature vector. As we can see, we can still obtain high-quality predictions in a more compact representation. Again, we obtained an accuracy of around 74% for *Diverse 1*, and 60% for *Diverse 2* towards the end of the runs.

6 Analysis of Coefficients

We analyze the coefficients that are learned in our experiments, in order to better understand the prediction functions. We focus here in analyzing the reduced feature vector in the Computer Go domain, as its lower number of coefficients makes the analysis more practicable (and its results are similar to the ones with the full feature vector). Hence, we study the different coefficients learned for each subset size (i.e., the size of the subset of agents that agrees on the chosen action), under different teams and board sizes. As a way to make a comparison among the coefficients under different situations, we plot the normalized coefficients, by dividing each by their overall sum. This allows us to study how the relative importance of each coefficient in determining whether a team will be successful changes under different situations. Similarly to the previous section, we also plot error bars showing the 99% confidence interval (i.e., $p = 0.01$) by performing a $t-test$ over the results of our random subsampling validation.

We start by plotting, in Figure 22, the coefficients organized according to the board sizes, to better compare how they change for each team. Note that the closer to 0 a coefficient is (i.e., the lower the bar size in the graphs), the higher the importance of the corresponding



(a) Accuracy

(b) Failure Precision

(c) Success Precision

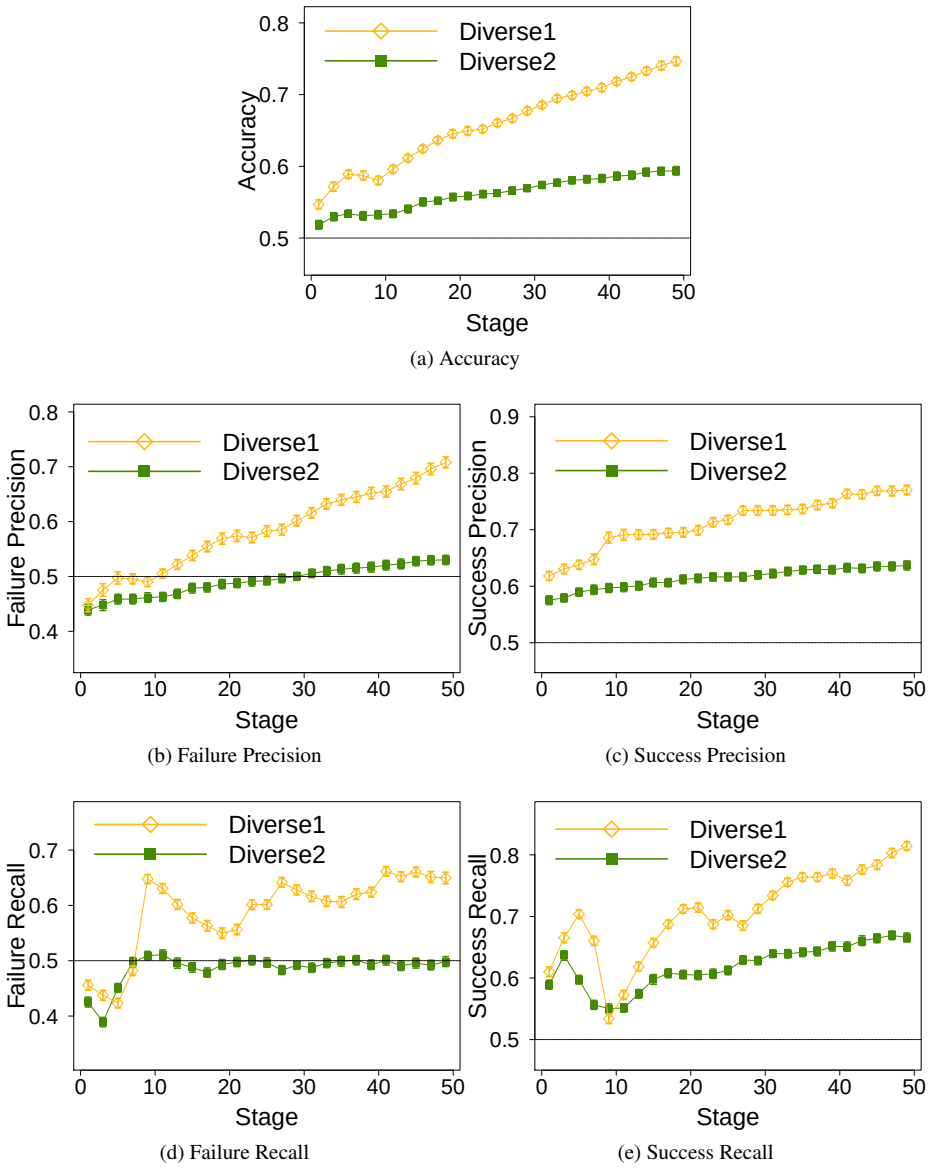
(d) Failure Recall

(e) Success Recall

Fig. 20: Performance metrics over a set of items in ensemble system classification (full feature vector).

subset size in determining that a game will be a victory for the team. Also, the numbers on the x axis indicate the respective subset size.

We can make several observations from these graphs. First, note that in general the learned coefficients are quite stable, as the low error bars indicate a low variance across different samples. Second, for almost all the coefficients we have a statistically significant



(a) Accuracy

(b) Failure Precision

(c) Success Precision

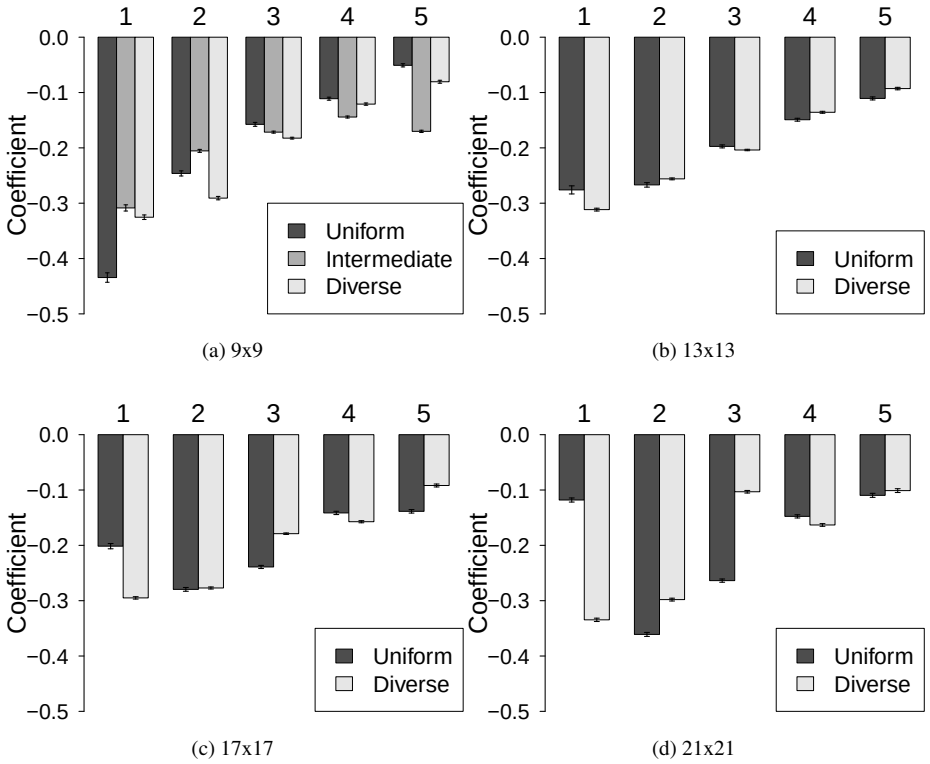
(d) Failure Recall

(e) Success Recall

Fig. 21: Performance metrics over a set of items in ensemble system classification (reduced feature vector).

difference across the different teams. This indicates that the learned functions are not trivial, as the appropriate values for each coefficient strongly depends on the team members.

Furthermore, we normally would expect the coefficients to increase (i.e., get closer to 0) as the respective subset size increases, capturing the notion that the higher the agreement, the higher the likelihood of success. However, that is not always the case, and we can, in fact, observe many surprising coefficients, especially in larger board sizes. On 21×21 Go,



(a) 9x9 (b) 13x13

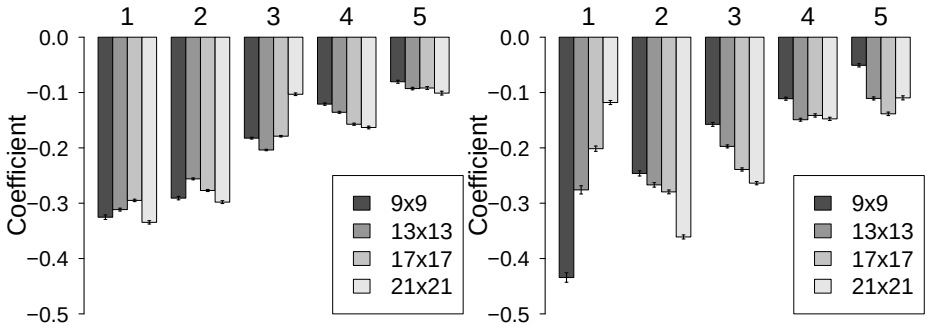
(c) 17x17 (d) 21x21

Fig. 22: Normalized coefficients of all teams and board sizes, organized by board sizes.

the "strict increase" hypothesis does not happen for both the *diverse* and the *uniform* teams. The coefficient for subsets of size 1 for the *uniform* team is the most surprising result, as it seems to indicate some tendency of winning the games when the agents fully disagree. For *intermediate*, we can notice such non-strict increase even on 9×9 Go (as the coefficient for subsets of size 4 is significantly higher than the one for subsets of size 5), which may be caused by this being the weakest team (and, hence, more agents agreeing on the same action may not really indicate a higher likelihood of the action being optimal).

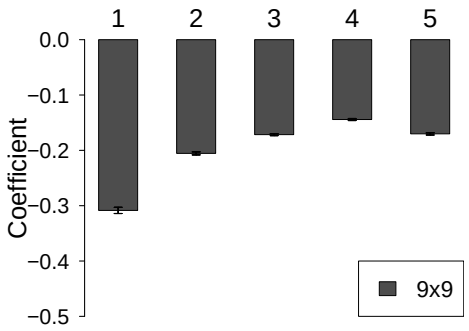
In order to better analyze how the coefficients change as the board size increases, we plot in Figure 23 the coefficients organized by teams. It is interesting to note that the coefficients evolve in different ways for the different teams. For *diverse*, we notice an increase in the importance of subsets of size 3 in 21×21 Go, curiously significantly surpassing subsets of size 4. One possible explanation could be that in such board size the team is more likely correct when the three strongest agents are in (exclusive) agreement.

For the *uniform* team the results are even more surprising. We can notice a consistent increase in the importance of subsets of size 1. This seems to indicate that the team is more likely to be winning when the team is in full disagreement as the board size grows, which does not correspond well with the *ST* agent model. The *ST* agent model, however, was originally created to model diverse teams [57], hence a deviation on uniform teams could be expected. Perhaps this result may happen because the games where the agents fully disagree



(a) Diverse Team

(b) Uniform Team



(c) Intermediate Team

Fig. 23: Normalized coefficients of all teams and board sizes, organized by teams.

tend to be more complex, and the *uniform* team may be in a higher advantage on dealing with such complex positions than its opponent.

It is worthy of notice that even though the prediction functions for these teams are so different, and evolve in such different ways as the number of actions increases, we could still learn them adequately for all these different situations, always obtaining high-quality prediction results. This shows that our learning methodology is robust, as it is able to learn good prediction functions for a variety of situations.

7 Discussion

We show, both theoretically and experimentally, that we can make high-quality predictions about the performance of a team of voting agents, using only information about the frequency of agreements among them. We present two kinds of feature vectors, one that includes information about which specific agents were involved in an agreement and one that only uses information about how many agents agreed. Although the number of features in the former increases exponentially with the number of agents, causing scalability concerns, the latter representation scales much better, as it increases linearly. Theoretically, the full feature vector should have better results in general, but as we discuss in Corollary 1, the reduced representation approximates well the full one under some conditions. In fact, however,

in our experiments we find that, although the results are similar, the reduced representation is statistically significantly better in most cases. This may happen since it is easier to learn a model with less features. Hence, for large teams we can safely use the reduced feature vector, avoiding scalability problems.

Moreover, in real applications we usually do not have extremely large teams of voting agents. Unless we have an "idealized" diverse team, the performance is expected to converge after a certain number of agents [38]. In Marcolino et al. (2013) [56] and Marcolino et al. (2014) [57], significant improvements are already obtained with only 6 agents, while Jiang et al. (2014) [38] show little improvement as teams grow larger than 15 agents. Therefore, even in cases where Corollary 1 does not apply, and the reduced vector is not a good approximation, the scalability of the full feature vector might not be a real concern.

Based on classical voting theory, and in our Proposition 1, Observation 1 and Proposition 2, we would expect the predictions to work better for the *uniform* team, if it is composed of copies of the best agent. However, we present a more general model in Theorem 1, that does not really depend on plurality being a maximum likelihood estimator (MLE). In fact, we show in our experiments that the prediction works significantly better for the *diverse* team, and we show theoretically that this can happen in our Observation 2. Moreover, the prediction for *intermediate* works as well as for the other teams, and also significantly better than for the other teams towards the last stages of the games (especially when playing against Pachi), even though it is a significantly weaker team. We would not expect this result, based on classical voting theory and Proposition 1, Observation 1 and Proposition 2. We can, however, expect such result based on our more general model in Theorem 1, as we show that the prediction does not really depend at all on plurality being a MLE. Hence, it can still work well in cases where the team is weaker, and the MLE assumption does not hold well.

We also observed that the quality of our prediction increases significantly as the board size increases, for both the *diverse* and the *uniform* team, but the impact for *diverse* occurs earlier in the game, and over a larger number of stages. We explain such phenomenon in Theorem 2. As normally complex problems are characterized by having a large action space, this shows that we can make better predictions for harder problems, when it is actually more useful to be able to make predictions about the team performance.

As we study the *Receiver Operating Characteristic* (ROC) curves, and the respective Areas Under the Curves (AUC), we can also show that our experimental conclusions hold irrespective of how we interpret the output of the prediction function. Hence, we find that across many different thresholds we can still make better predictions for the *diverse* team, and the prediction quality still improves in larger boards. We also notice that the difference on the prediction quality for *diverse* and *uniform* teams increases in larger boards.

We analyze the coefficients of the reduced feature vector, and note that they are highly non-trivial. The way the coefficient for subsets of size 1 evolves for the *uniform* team is the most surprising result, and seems to suggest that the performance improves for such team for a different reason than the one presented in Theorem 2. As the theorem assumes *ST* agents, however, we can expect it to cover better the situation of the *diverse* team. A better understanding of why the coefficients turned out to be how they are is an interesting direction for future work. Nevertheless, it is a positive result that our learning technique could work well and learn adequate prediction functions across all these different situations.

We also applied our technique to the Ensemble Learning domain, where we were able to predict the final performance of a team of classifiers in a dataset of hand-written digits recognition. This shows that our model is really domain independent, as it worked well in two completely different domains.

However, although we showed a great performance in prediction, we did not present in this paper what an operator should actually do as the prediction of failure goes high. Possible remedial procedures (and the relative qualities of each one of them) vary according to each domain, but here we discuss some possible situations (as a reminder, voting has been applied in many different domains, such as crowdsourcing [55,4], board games [56,57,64], machine learning [66], forecasting systems [37], etc, and hence there are many potential domains for our technique).

For example, consider a complex problem being solved in a cluster of computers. We do not want to overly allocate resources, as that would be a waste of computational power that could be allocated to other tasks (or a waste of electrical energy, at the very least). However, we could increase the allocation of resources for solving the current problem when it becomes necessary, according to our prediction.

Moreover, it is known that the best voting rule depends on the noise model of the agents [19]. However, in general, such model is not known for existing agents [56]. Therefore, we could start by using a very general rule, such as plurality voting, and dynamically try different voting rules according to our current prediction. Note that although Proposition 1, Observation 1 and Proposition 2 refer to plurality voting, our general model presented in Theorem 1 does not really depend on the voting rule.

Additionally, the best team may depend on which specific problem we are facing. Hence, we could start by using the team that generally works the best, and dynamically change it according to our prediction, in order to adapt to the current problem. For example, while playing games (such as board games like Go, or card games like Poker, or real time strategy games like Spring, etc), it is well known that the best strategy changes according to each specific opponent we are facing [29,72,50,70,5]. However, it is in general hard to know which player is our current antagonist. Therefore, we could start playing the game with the team that works the best against general opponents, and dynamically change the team members or the individual agent's strategies as our prediction of failure goes high, trying to adapt to the current situation.

Note, however, that this requires the learned functions to generalize beyond the training conditions. Training and testing under different conditions leads to a transfer learning problem, which is a current topic of research in the machine learning literature [6,45,75]. In this paper we explored the situation of training against one adversary, but testing against another. In our case we were still able to obtain good results when testing against a different adversary, surprisingly in some situations even better than when testing against the same adversary. This shows the generality of the learned functions. However, exploring transfer learning in the context of our technique is still an interesting direction for future work, as we may need to switch team members for failure recovery – which requires reusing and adapting learned functions across different teams.

8 Conclusion

Voting is a widely applied domain independent technique, that has been used in a variety of domains, such as: machine learning, crowdsourcing, board games, forecasting systems, etc. In this paper, we present a novel method to predict the performance of a team of agents that vote together at every step of a complex problem. Our method does not use any domain knowledge and is based only on the frequencies of agreement among the agents.

We explain theoretically why our prediction works. First, we present an explanation based on classical voting theories, but such explanation fails to fully explain our experimen-

tal results. Hence, we also present a more general model, that is independent of the voting rule, and also independent of any optimality assumptions about the voting rule (i.e., the voting rule does not need to be a maximum likelihood estimator across all world states). Such model allows a deeper understanding of the prediction methodology, and a more complete comprehension of our experimental results.

We perform experiments in the Computer Go domain with three different teams, each having different levels of diversity and strength (i.e., performance), four different board sizes, and two different adversaries. We showed that the prediction works online at each step of the problem solving process, and matches often with an in-depth analysis (that takes orders of magnitude longer time). Hence, we could achieve a high prediction quality for all teams, despite their differences. In particular, we show that, contrary to what would be expected based on classical voting models, our prediction is not better for stronger teams, and can actually work equally well (even significantly better in some cases) for a very weak team. Moreover, we showed that we could achieve a higher prediction quality for a diverse team than for a uniform team. Furthermore, we verified experimentally that the prediction quality increase as the action space (board size) grows (and the impact occurs earlier in the game for the diverse team). All that could be expected based on our more general model. Additionally, we showed experimentally that the technique still obtains good results when testing against a different adversary than in the training condition, demonstrating the generality of the trained functions.

We also study the *Receiver Operating Characteristic* curves of our results, and their respective Areas Under the Curves (AUC). Hence, we verified that our conclusions hold across many different thresholds (i.e., different ways to interpret the output of our prediction function).

Moreover, we tested our approach when predicting the final performance of a team of classifiers, a domain that is very different than Computer Go. We were still able to obtain high-quality predictions, showing that our approach really works across different domains.

Finally, we studied in detail the prediction functions learned. Our analysis showed that these functions are highly non-trivial, and vary significantly according to each team and board size. In fact, better understanding how and why these coefficients evolved in the way they did as the board size grows for different teams is an open direction for further study. Overall, however, our prediction technique is very robust, as it could learn adequate functions across all these situations.

Therefore, a system operator can use our technique to take remedial procedures if the team is not performing well, or incorporated within an automatic procedure to dynamically change the team and/or the voting rule. We discussed in detail how that could be executed in different domains.

Hence, this work is a significant step towards not only a deeper understanding of voting systems, but also towards a greater applicability of voting in a variety of domains, by combining the potential of voting in finding correct answers with the ability to access the performance of teams and the predictability of the final outcome.

Acknowledgements This research was supported by MURI grant W911NF-11-1-0332, and by IUSSTF. The authors would like to thank Chao Zhang and Amulya Yadav for useful comments.

References

1. AL-Malaise, A., Malibari, A., Alkhozae, M.: Students' performance prediction system using multi agent data mining technique. International Journal of Data Mining & Knowledge Management Process **4**(5), 1–20 (2014)
2. Amanatidis, G., Barrot, N., Lang, J., Markakis, E., Ries, B.: Multiple referenda and multiwinner elections using hamming distances: Complexity and manipulability. In: Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems, AAMAS, pp. 715–723 (2015)
3. Aziz, H., Gaspers, S., Gudmundsson, J., Mackenzie, S., Mattei, N., Walsh, T.: Computational aspects of multi-winner approval voting. In: Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems, AAMAS, pp. 107–115 (2015)
4. Bachrach, Y., Graepel, T., Kasneci, G., Kosinski, M., Van Gael, J.: Crowd IQ: aggregating opinions to boost performance. In: Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems, AAMAS, pp. 535–542 (2012)
5. Bakkes, S., Spronck, P., van den Herik, J.: Opponent modelling for case-based adaptive game AI. Entertainment Computing **1**(1), 27–37 (2009)
6. Banerjee, B., Stone, P.: General game learning using knowledge transfer. In: Proceedings of the 20th International Joint Conference on Artificial Intelligence, pp. 672–677 (2007)
7. Bartholdi III, J.J., Tovey, C.A., Trick, M.A.: The computational difficulty of manipulating an election. Social Choice and Welfare **6**(3), 227–241 (1989)
8. Bartholdi III, J.J., Tovey, C.A., Trick, M.A.: Voting schemes for which it can be difficult to tell who won the election. Social Choice and Welfare **6**(3), 157–165 (1989)
9. Baudiš, P., Gailly, J.l.: Pachi: State of the Art Open Source Go Program. In: Advances in Computer Games 13, pp. 24–38 (2011)
10. Bialkowski, A., Lucey, P., Carr, P., Yue, Y., Sridharan, S., Matthews, I.: Large-scale analysis of soccer matches using spatiotemporal tracking data. In: Proceedings of the 2014 IEEE International Conference on Data Mining, ICDM, pp. 725–730 (2014)
11. Brandl, F., Brandt, F., Hofbauer, J.: Incentives for participation and abstention in probabilistic social choice. In: Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems, AAMAS, pp. 1411–1420 (2015)
12. Browne, C., Powley, E.J., Whitehouse, D., Lucas, S.M., Cowling, P.I., Rohlfshagen, P., Tavener, S., Liebana, D.P., Samothrakis, S., Colton, S.: A survey of monte carlo tree search methods. IEEE Transactions on Computational Intelligence and AI in Games **4**(1), 1–43 (2012)
13. Bulling, N., Dastani, M., Knobbout, M.: Monitoring norm violations in multi-agent systems. In: Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems, AAMAS, pp. 491–498 (2013)
14. Caragiannis, I., Procaccia, A.D., Shah, N.: When do noisy votes reveal the truth? In: Proceedings of the 14th ACM conference on Electronic commerce, EC, pp. 143–160. ACM, New York, NY, USA (2013)
15. Cesa-Bianchi, N., Lugosi, G.: Prediction, Learning, and Games. Cambridge University Press (2006)
16. Chatterji, S., Sen, A., Zeng, H.: Random dictatorship domains. Games and Economic Behavior **86**, 212–236 (2014)
17. Chiu, B.C., Webb, G.I.: Using decision trees for agent modeling: Improving prediction performance. User Modeling and User-Adapted Interaction **8**, 131–152 (1998)
18. Conitzer, V., Sandholm, T.: Universal voting protocol tweaks to make manipulation hard. In: Proceedings of the 18th International Joint Conference on Artificial Intelligence, IJCAI, pp. 781–788 (2003)
19. Conitzer, V., Sandholm, T.: Common voting rules as maximum likelihood estimators. In: Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence, UAI, pp. 145–152. Morgan Kaufmann Publishers (2005)
20. Cortes, C., Vapnik, V.: Support-vector networks. Machine Learning **20**(3), 273–297 (1995)
21. Cover, T.M., Hart, P.E.: Nearest neighbor pattern classification. IEEE Transactions on Information Theory **13**(1), 21–27 (1967)
22. Cox, D.R.: The regression analysis of binary sequences (with discussion). Journal of the Royal Statistical Society B **20**, 215–242 (1958)
23. Davies, J., Katsirelos, G., Narodytska, N., Walsh, T.: Complexity of and algorithms for Borda manipulation. In: Proceedings of the 25th AAAI Conference on Artificial Intelligence, pp. 657–662 (2011)
24. Doan, T.T., Yao, Y., Alechina, N., Logan, B.: Verifying heterogeneous multi-agent programs. In: Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems, AAMAS, pp. 149–156 (2014)
25. Elkind, E., Shah, N.: Electing the most probable without eliminating the irrational: Voting over intransitive domains. In: Proceedings of the 30th Conference on Uncertainty in Artificial Intelligence, UAI, pp. 182–191 (2014)

26. Enzenberger, M., Müller, M., Arneson, B., Segal, R.: Fuego - An open-source framework for board games and Go engine based on Monte Carlo Tree Search. IEEE Transactions on Computational Intelligence and AI in Games **2**(4), 259 –270 (2010)
27. Free Software Foundation: Gnugo. `http://www.gnu.org/software/gnugo/` (2009)
28. Fu, B., Wang, Z., Pan, R., Xu, G., Dolog, P.: An integrated pruning criterion for ensemble learning based on classification accuracy and diversity. In: L. Uden, F. Herrera, J.B. Perez, J.M.C. Rodriguez (eds.) Proceedings of the 7th International Conference on Knowledge Management in Organizations: Service and Cloud Computing, KMO'12, *Advances in Intelligent Systems and Computing*, vol. 172, pp. 47–58. Springer (2012)
29. Ganzfried, S., Sandholm, T.: Game theory-based opponent modeling in large imperfect-information games. In: Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems, AAMAS, pp. 533–540 (2011)
30. Gelly, S., Wang, Y., Munos, R., Teytaud, O.: Modification of UCT with patterns in Monte-Carlo Go. Tech. rep., Institut National de Recherche en Informatique et en Automatique (2006)
31. Guttmann, C.: Making allocations collectively: Iterative group decision making under uncertainty. In: R. Bergmann, G. Lindemann, S. Kirn, M. Pechoucek (eds.) Proceedings of the 6th German Conference on Multiagent System Technologies, *Lecture Notes in Computer Science*, vol. 5244, pp. 73–85. Springer, Kaiserslautern, Germany (2008)
32. Hand, D.J., Till, R.J.: A simple generalization of the area under the ROC curve for multiple class classification problems. Machine Learning **45**, 171–186 (2001)
33. Hanley, J.A., McNeil, B.J.: The meaning and use of the area under a receiver operating characteristic (ROC) curve. Radiology **143**(1), 29–36 (1982)
34. Heiny, E.L., Blevins, D.: Predicting the atlanta falcons play-calling using discriminant analysis. Journal of Quantitative Analysis in Sports **7**(3), 1–12 (2011)
35. Hong, L., Page, S.E.: Groups of diverse problem solvers can outperform groups of high-ability problem solvers. Proceedings of the National Academy of Sciences of the United States of America **101**(46), 16,385–16,389 (2004)
36. Hunter, J., Raimondi, F., Rungta, N., Stocker, R.: A synergistic and extensible framework for multi-agent system verification. In: Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems, AAMAS, pp. 869–876 (2013)
37. Isa, I.S., Omar, S., Saad, Z., Noor, N., Osman, M.: Weather forecasting using photovoltaic system and neural network. In: Proceedings of the Second International Conference on Computational Intelligence, Communication Systems and Networks, CICSyN, pp. 96–100 (2010)
38. Jiang, A.X., Marcolino, L.S., Procaccia, A.D., Sandholm, T., Shah, N., Tambe, M.: Diverse randomized agents vote to win. In: Proceedings of the 2014 Neural Information Processing Systems Conference, NIPS, pp. 2573–2581 (2014)
39. Kalech, M., Kaminka, G.A.: On the design of coordination diagnosis algorithms for teams of situated agents. Artificial Intelligence **171**, 491–513 (2007)
40. Kalech, M., Kaminka, G.A.: Coordination diagnostic algorithms for teams of situated agents: Scaling-up. Computational Intelligence **27**(3), 393–421 (2011)
41. Kaminka, G.A.: Coordination of Large-Scale Multiagent Systems, chap. Handling Coordination Failures in Large-Scale Multi-Agent Systems, pp. 273–286. Springer (2006)
42. Kaminka, G.A., Tambe, M.: What is wrong with us? Improving robustness through social diagnosis. In: Proceedings of the National Conference on Artificial Intelligence, AAAI, pp. 97–105 (1998)
43. Kass, G.V.: An exploratory technique for investigating large quantities of categorical data. Applied Statistics **29**(2), 119–127 (1980)
44. Khalastchi, E., Kalech, M., Rokach, L.: A hybrid approach for fault detection in autonomous physical agents. In: Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems, AAMAS, pp. 941–948 (2014)
45. Konidaris, G., Scheidwasser, I., Barto, A.G.: Transfer in reinforcement learning via shared features. The Journal of Machine Learning Research **13**(1), 1333–1371 (2012)
46. Kouvaros, P., Lomuscio, A.: Automatic verification of parameterised interleaved multi-agent systems. In: In Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems, AAMAS, pp. 861–868 (2013)
47. LiCalzi, M., Surucu, O.: The power of diversity over large solution spaces. Management Science **58**(7), 1408–1421 (2012)
48. Lindner, M.Q., Agmon, N.: Effective, quantitative, obscured observation-based fault detection in multi-agent systems. In: Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems, AAMAS, pp. 1449–1450 (2014)
49. List, C., Goodin, R.E.: Epistemic democracy: Generalizing the Condorcet Jury Theorem. Journal of Political Philosophy **9**, 277–306 (2001)

50. Lockett, A.J., Chen, C.L., Miikkulainen, R.: Evolving explicit opponent models in game playing. In: Proceedings of the 9th annual conference on genetic and evolutionary computation, GECCO, pp. 2106–2113 (2007)
51. Lucey, P., Bialkowski, A., Carr, P., Yue, Y., Matthews, I.: "How to get an open shot": Analyzing team movement in basketball using tracking data. In: Proceedings of the 8th Annual MIT SLOAN Sports Analytics Conference (2014)
52. Lucey, P., Bialkowski, A., Monfort, M., Carr, P., Matthews, I.: "Quality vs quantity": Improved shot prediction in soccer using strategic features from spatiotemporal data. In: Proceedings of the 9th Annual MIT SLOAN Sports Analytics Conference (2015)
53. Maheswaran, R., Chang, Y., Henehan, A., Danesis, S.: Deconstructing the rebound with optical tracking data. In: Proceedings of the 6th Annual MIT SLOAN Sports Analytics Conference (2012)
54. Manning, C.D., Raghavan, P., Schtze, H.: Introduction to Information Retrieval. Cambridge University Press (2008)
55. Mao, A., Procaccia, A.D., Chen, Y.: Better Human Computation Through Principled Voting. In: Proceedings of the 27th AAAI Conference on Artificial Intelligence, AAAI, pp. 1142–1148 (2013)
56. Marcolino, L.S., Jiang, A.X., Tambe, M.: Multi-agent team formation: Diversity beats strength? In: Proceedings of the 23rd International Joint Conference on Artificial Intelligence, IJCAI, pp. 279–285 (2013)
57. Marcolino, L.S., Xu, H., Jiang, A.X., Tambe, M., Bowring, E.: Give a hard problem to a diverse team: Exploring large action spaces. In: Proceedings of the 28th AAAI Conference on Artificial Intelligence, AAAI, pp. 1485–1491 (2014)
58. Marcolino, L.S., Zhang, C., Jiang, A.X., Tambe, M.: A detailed analysis of a multi-agent diverse team. In: T. Balke, A. Chopra, F. Dignum, B. van Riemsdijk (eds.) Coordination, Organizations, Institutions and Norms in Agent Systems IX, Lecture Notes in AI, pp. 3–24. Springer-Verlag (2014)
59. Mason, S.J., Graham, N.E.: Areas beneath the relative operating characteristics (ROC) and relative operating levels (ROL) curves: Statistical significance and interpretation. Quarterly Journal of the Royal Meteorological Society **128**, 2145–2166 (2002)
60. Mirchevska, V., Luštrek, M., Bežek, A., Gams, M.: Discovering strategic behaviour of multi-agent systems in adversary settings. Computing and Informatics **33**(1), 79–108 (2014)
61. Nagarajan, V., Marcolino, L.S., Tambe, M.: Every team deserves a second chance: Identifying when things go wrong. In: Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems, AAMAS, pp. 695–704 (2015)
62. Nair, R., Tambe, M.: Hybrid BDI-POMDP framework for multiagent teaming. Journal of Artificial Intelligence Research **23**(1), 367–420 (2005)
63. Nurmi, H.: Comparing Voting Systems. Springer (1987)
64. Obata, T., Sugiyama, T., Hoki, K., Ito, T.: Consultation algorithm for Computer Shogi: Move decisions by majority. In: Computer and Games'10, *Lecture Notes in Computer Science*, vol. 6515, pp. 156–165. Springer (2011)
65. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)
66. Polikar, R.: Ensemble Machine Learning: Methods and Applications, chap. Ensemble Learning, pp. 1–34. Springer (2012)
67. Quenzel, J., Shea, P.: Predicting the winner of tied NFL games: Do the details matter? Journal of Sports Economics (2014). Available online
68. Raines, T., Tambe, M., Marsella, S.: Automated assistants to aid humans in understanding team behaviors. In: AGENTS, pp. 419–426 (2000)
69. Ramos, F., Ayanegui, H.: Discovering tactical behavior patterns supported by topological structures in soccer-agent domains. In: Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems, AAMAS, pp. 1421–1424 (2008)
70. Schadd, F., Bakkes, S., Spronck, P.: Opponent modeling in real-time strategy games. In: Proceedings of the 2007 Simulation and AI in Games Conference, GAMEON, pp. 61–68 (2007)
71. Soejima, Y., Kishimoto, A., Watanabe, O.: Evaluating root parallelization in Go. IEEE Transactions in Computational Intelligence and AI in Games **2**(4), 278–287 (2010)
72. Southey, F., Bowling, M., Larson, B., Piccione, C., Burch, N., Billings, D., Rayner, C.: Bayes' bluff: Opponent modeling in poker. In: Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence, UAI, pp. 550–558 (2005)
73. Sylvester, J., Chawla, N.V.: Evolutionary ensembles: Combining learning agents using genetic algorithms. In: AAAI Workshop on Multiagent Learning, AAAI (2005)

74. Tarapore, D., Christensen, A.L., Lima, P.U., Carneiro, J.: Abnormality detection in multiagent systems inspired by the adaptive immune system. In: Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems, AAMAS, pp. 23–30 (2013)
75. Taylor, M.E., Stone, P.: Transfer learning for reinforcement learning domains: A survey. The Journal of Machine Learning Research **10**, 1633–1685 (2009)
76. Torrey, L., Taylor, M.E.: Teaching on a budget: Agents advising agents in reinforcement learning. In: Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems, AAMAS, pp. 1053–1060 (2013)
77. Xia, L., Conitzer, V.: Determining possible and necessary winners under common voting rules given partial orders. Journal of Artificial Intelligence Research (JAIR) **41**, 25–67 (2011)
78. Xia, L., Conitzer, V.: A maximum likelihood approach towards aggregating partial orders. In: Proceedings of the 22nd International Joint Conference on Artificial Intelligence, IJCAI, pp. 446–451 (2011)
79. Yang, Y.: Manipulation with bounded single-peaked width: A parameterized study. In: Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems, AAMAS, pp. 77–86 (2015)
80. Zhang, C., Lesser, V.: Coordinating multi-agent reinforcement learning with limited communication. In: Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems, AAMAS, pp. 1101–1108 (2013)

A Alternative Baseline

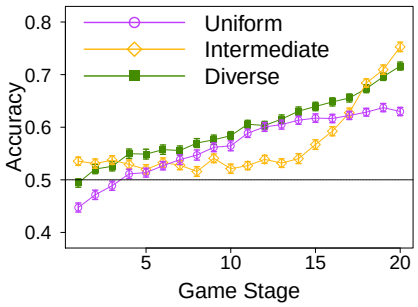
In this section we present our results when comparing with the actual final outcome of the games, instead of using Fuego's evaluation. We show these results because some readers could be interested in seeing this alternative evaluation methodology.

In Figure 24 we show the accuracy, precision and recall for 9x9 Go games, while in Figure 25 we show the ROC curves for such games. In Figure 26 we can see the ROC results for all the 4 different board sizes, while we plot the AUC results with one graph per team in Figure 27. We also show the AUC results with one graph per board size in Figure 28. In Figure 29 we plot the difference between the areas under the AUC graphs for *diverse* and *uniform*, and also the percentage of stages where the prediction for *diverse* is significantly better. Finally, in Figure 30 we show the accuracy, precision and recall for 21×21 Go (and also 9×9 Go, for comparison). As we can see, the results are similar to the previous ones, and our main conclusions still hold: we still can make better predictions for the *diverse* team than for the *uniform* team, and we can observe a better quality when predicting for 21×21 Go than smaller board sizes. We notice, however, that this time the difference between *diverse* and *uniform* (both in terms of area under the AUC curves and percentage of stages where the AUC for *diverse* is significantly better) is higher on 9×9 Go than on 21×21 Go.

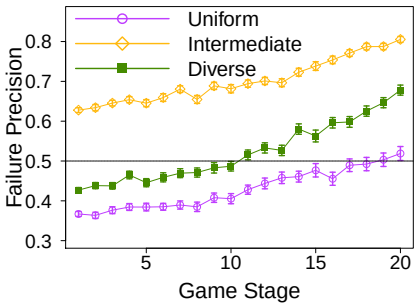
We compare the full and the reduced representation under this alternative baseline as well. In Figure 31 we show the area under the AUC curves for all teams and board sizes under consideration. Again, the results of both representations are similar[5], but the reduced one is significantly better in almost all cases.

Finally, we show the results when training against Pachi (dashed line) or Fuego (continuous line), and testing against Pachi. In Figure 32 we show the accuracy, precision, and recall results. In Figure 33 we show the ROC curves, and in Figure 34 the AUC curves. We also compare the full and the reduced representations in Figure 35. As we can see, the results are similar to the ones presented in the main paper, and our conclusions still hold.

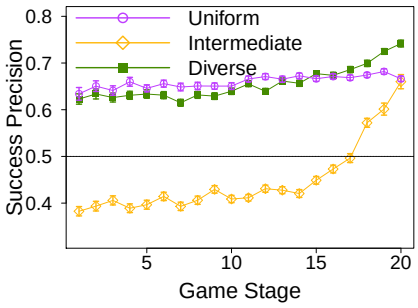
[5] Our main results still hold, except that the AUC for 21×21 is significantly better than for 9×9 earlier for *uniform* than *diverse* under the reduced representation. For *uniform*, since stage 12, while for *diverse*, since stage 15.



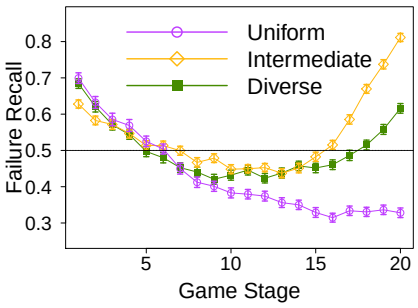
(a) Accuracy



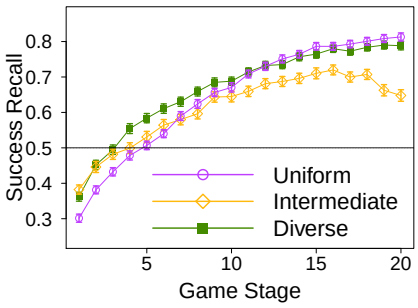
(b) Failure Precision



(c) Success Precision

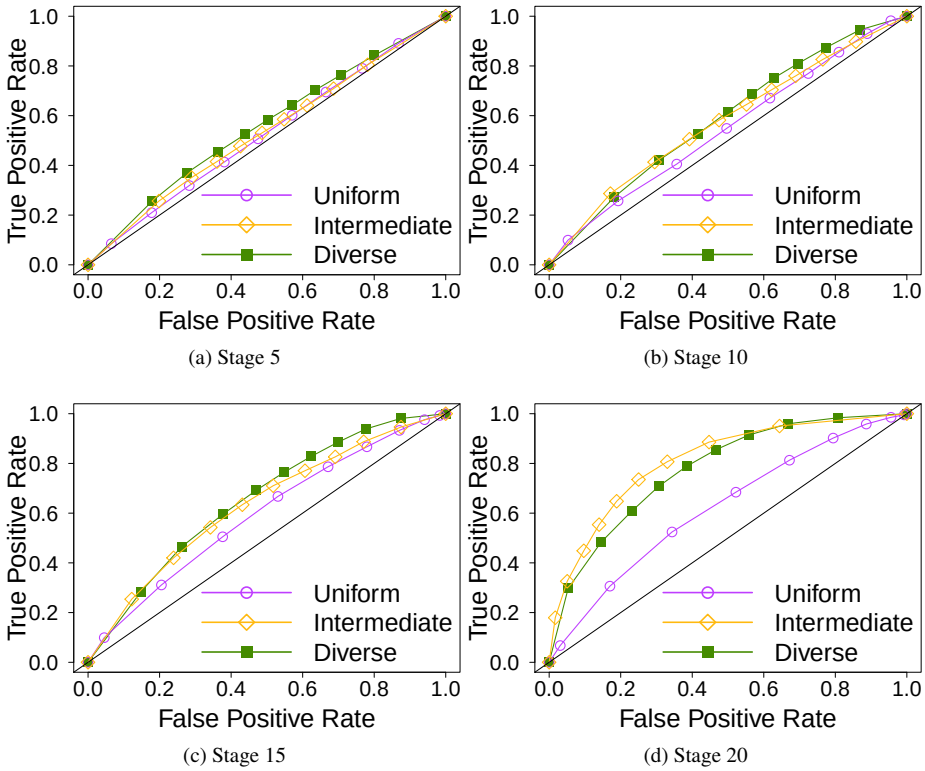


(d) Failure Recall



(e) Success Recall

Fig. 24: Performance metrics over all turns of 9x9 Go games. (Alternative baseline)



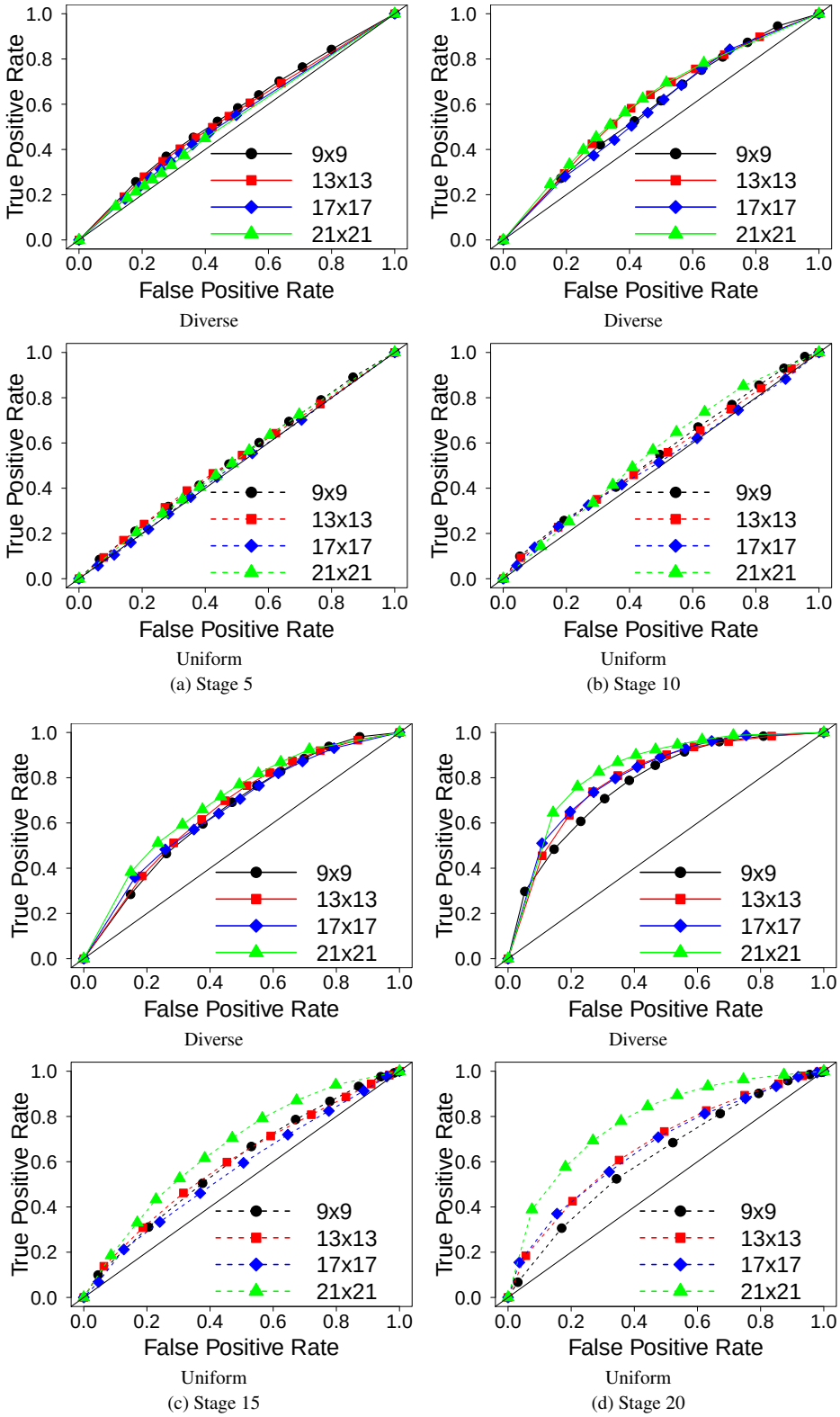
(a) Stage 5

(b) Stage 10

(c) Stage 15

(d) Stage 20

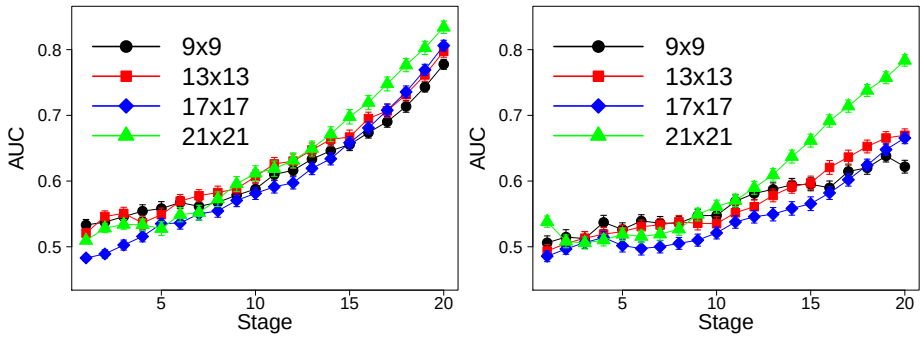
Fig. 25: ROC curves, analyzing different thresholds in 9x9 Go. (Alternative baseline)



(a) Stage 5 (b) Stage 10

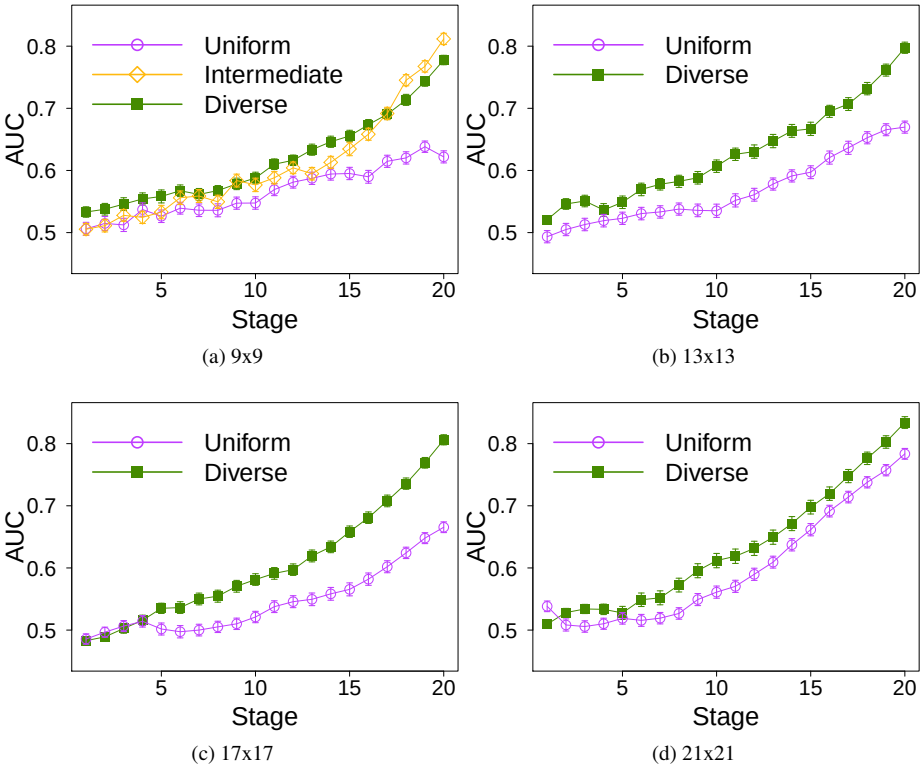
(c) Stage 15 (d) Stage 20

Fig. 26: ROC curves for *diverse* and *uniform*, for different board sizes. (Alternative baseline)



(a) Diverse Team (b) Uniform Team

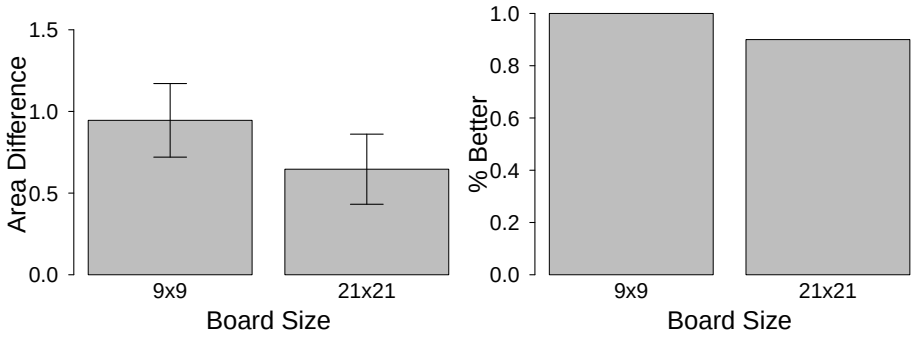
Fig. 27: AUC for different teams and board sizes, organized by teams. (Alternative baseline)



(a) 9x9 (b) 13x13

(c) 17x17 (d) 21x21

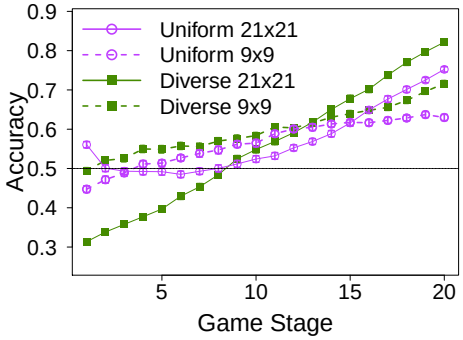
Fig. 28: AUC for different teams and board sizes, organized by board sizes. (Alternative baseline)



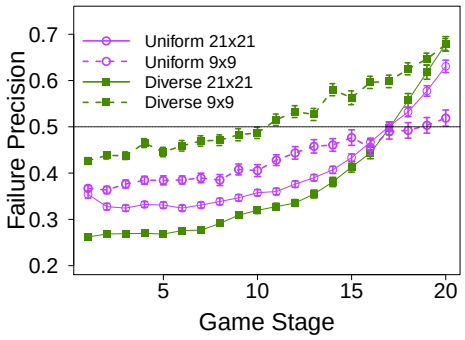
(a) By area

(b) By percentage of stages

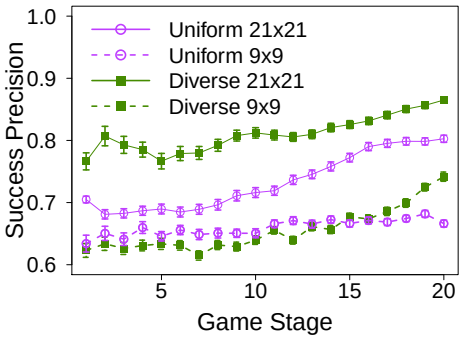
Fig. 29: Differences in prediction quality for the *diverse* and *uniform* teams. (Alternative baseline)



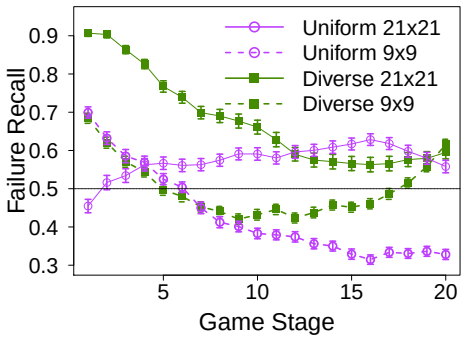
(a) Accuracy



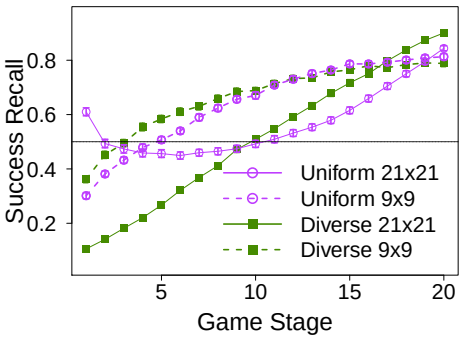
(b) Failure Precision



(c) Success Precision



(d) Failure Recall



(e) Success Recall

Fig. 30: Performance metrics over all turns of 9x9 and 21x21 Go games. (Alternative baseline)

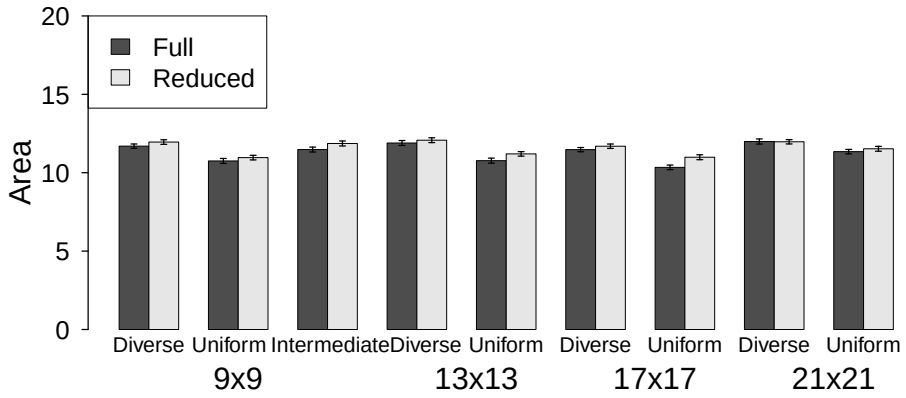
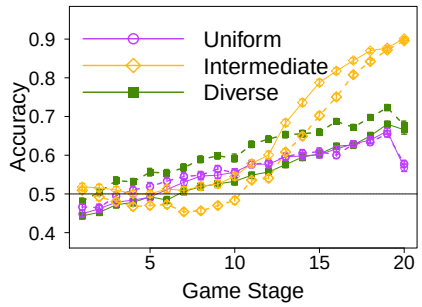
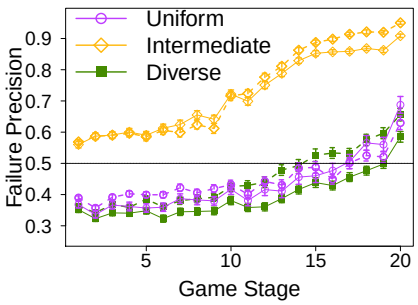


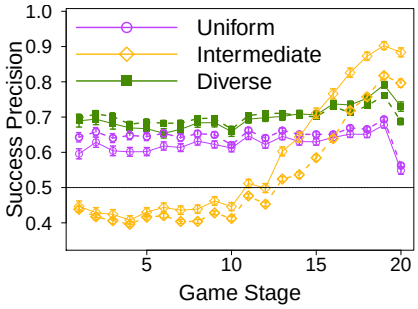
Fig. 31: Comparison of prediction quality with the full and reduced representation. (Alternative baseline)



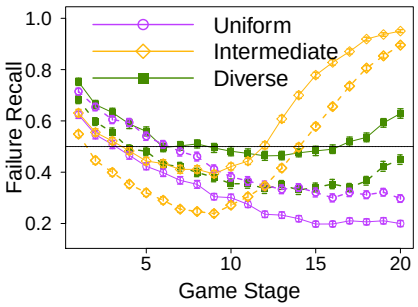
(a) Accuracy



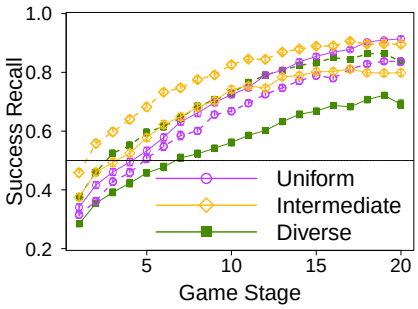
(b) Failure Precision



(c) Success Precision

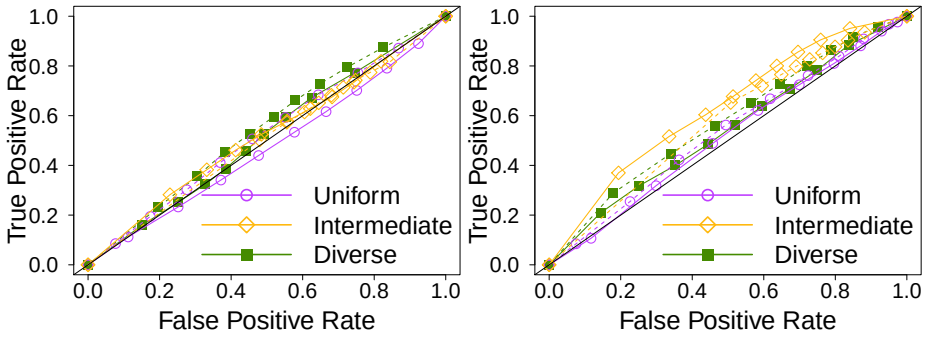


(d) Failure Recall



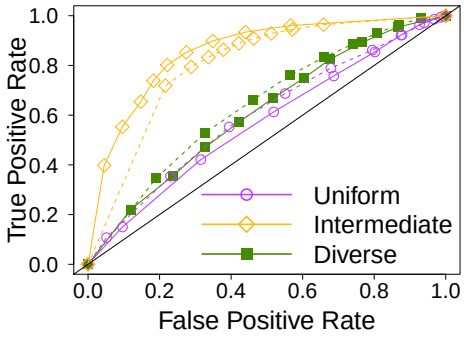
(e) Success Recall

Fig. 32: Performance metrics over all turns of 9x9 Go games, when training against Pachi (dashed line) or Fuego (continuous line), and testing against Pachi. (Alternative baseline)

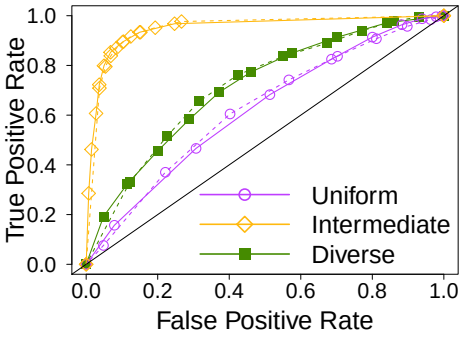


(a) Stage 5

(b) Stage 10



(c) Stage 15



(d) Stage 20

Fig. 33: ROC curves, analyzing different thresholds in 9x9 Go, when training against Pachi (dashed line) or Fuego (continuous line), and testing against Pachi. (Alternative baseline)

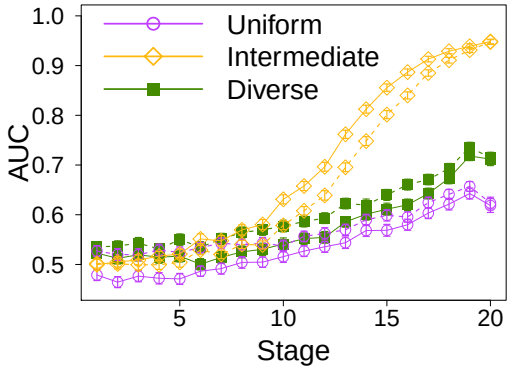
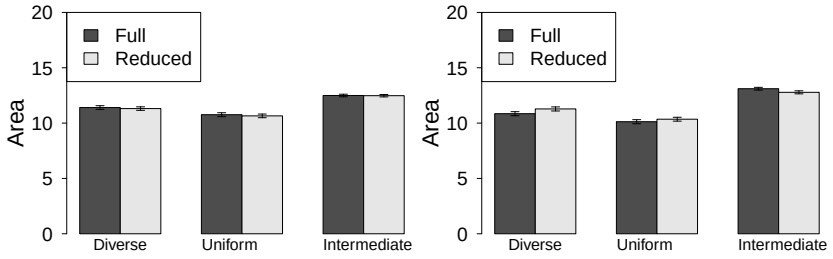


Fig. 34: AUC for *diverse*, *uniform* and *intermediate* teams, in 9x9 Go, when training against Pachi (dashed line) or Fuego (continuous line), and testing against Pachi. (Alternative baseline)



(a) Training Against Pachi (b) Training Against Fuego

Fig. 35: Comparison of prediction quality with the full and reduced representation, when testing against Pachi. (Alternative baseline)