Journal of Artificial Intelligence Research 73 (2022) 277-327 Submitted 04/2021; published 1/2022

A Survey on Opponent Modeling in Adversarial Domains

Samer B. Nashed SNASHED@CS.UMASS.EDU
Shlomo Zilberstein SHLOMO@CS.UMASS.EDU
University of Massachusetts Amherst
Manning College of Information and Computer Sciences
Amherst, MA 01002 USA

Abstract

Opponent modeling is the ability to use prior knowledge and observations in order to predict the behavior of an opponent. This survey presents a comprehensive overview of existing opponent modeling techniques for adversarial domains, many of which must address stochastic, continuous, or concurrent actions, and sparse, partially observable payoff structures. We discuss all the components of opponent modeling systems, including feature extraction, learning algorithms, and strategy abstractions. These discussions lead us to propose a new form of analysis for describing and predicting the evolution of game states over time. We then introduce a new framework that facilitates method comparison, analyze a representative selection of techniques using the proposed framework, and highlight common trends among recently proposed methods. Finally, we list several open problems and discuss future research directions inspired by AI research on opponent modeling and related research in other disciplines.

1. Introduction

The goal of this survey is four-fold. First, we present a comprehensive review of work on opponent modeling in adversarial domains. Second, we present a new mathematical lens through which to understand and analyze the interactions of two or more strategies or policies. Third, we introduce a common framework by which to compare and evaluate opponent modeling techniques. And fourth, we discuss open problems in opponent modeling and areas for potential research, given the current state-of-the-art.

Work on opponent modeling has been used in a number of real-world applications, including professional sports such as football, basketball, and tennis, military tasks such as formation prediction and patrol behavior, and even video game design (Ontanón et al., 2013; Bakkes et al., 2012). These techniques have also been applied to a variety of basic research initiatives, robot soccer and its variants chief among them. In theory, many of these domains can be modeled using Partially Observable Stochastic Games (POSGs), and under certain conditions POSG solvers can find provably optimal strategies. However, many scenarios of interest are either so complex that the resultant POSGs become intractable, or do not have payoff functions that can be written down, either because they are too complicated or because they are simply unknown. These types of scenarios require a different approach. Instead of modeling the problem completely and computing the optimal strategy *a priori*, opponent modeling research uses data gathered from past experience, or even online, to complete or refine models of opponent behavior which are only partially predefined. Additionally, some interactions may support multiple incompatible equilibria, and detecting which actions are compatible with the current equilibrium requires opponent modeling.

The academic literature has cited many potential applications for opponent modeling research, and in an effort to simplify discussion and comparison of different techniques and concepts we will use a running example. RoboCup (Kitano et al., 1997) is an international robot soccer competition held annually, in which teams of robots compete against each other in games similar to the sport of football (soccer) played by humans. Although RoboCup and its various divisions exhibit differences from other opponent modeling applications, it retains many of the fundamental properties that make it a challenging and useful domain.

Principally, these properties include continuous actions, stochastic actions, concurrent actions, a sparse and partially observable payoff structure, and in some RoboCup leagues, decentralized coordination of multiple agents. For example, the drive goals sent to robots are continuous position or velocity variables (x, y, θ) and $(\dot{x}, \dot{y}, \dot{\theta})$, respectively. Desired directions and velocities for shots and passes are also continuous variables. Moreover, when robots execute these drive or kick actions, the outcomes are stochastic, described by a probability density function over real-valued robot or ball position and velocity variables at some future time. Because shots where a goal is scored are the only event with a direct effect on the victory condition, the payoff structure is sparse, with most actions having zero immediate payoff for either team. Additionally, even if an evaluation function that produces payoff estimates for non-goal actions is available, the optimal form of such a function is not generally agreed upon, and agents do not know the functions their opponents employ.

Unsurprisingly, these properties have attracted the attention of many researchers studying opponent modeling, and the RoboCup simulation league in particular has supported a variety of opponent modeling research (Pourmehr & Dadkhah, 2011). From here on we will illustrate concepts using RoboCup-based examples, and because of this grounding we will refer to the set of variables relevant to decision making and prediction, both fully and partially observable, as the 'game state', Y. Although most of the research presented in this survey does not study problems exhibiting all of the above properties simultaneously, for instance some approaches assume discrete actions, deterministic actions, turn-based play, etc., we nonetheless believe there is valuable perspective in studying how problem formulations evolve to exploit additional information or certainty along various dimensions of the problem. Some of this research is also discussed by Albrecht & Stone (2018), although we focus more heavily on adversarial domains and in particular on how various approaches model the evolution of the game state.

The high degree of specialization required for individual opponent modeling approaches to work well in practice often obscures trends and patterns present in the literature. Furthermore, as individual papers focus on domain-specific aspects of their application, it can be difficult to identify chronic limitations or uncover implicit assumptions within opponent modeling research more broadly. We feel there is potential for substantial advance in opponent modeling research, especially as unsupervised learning techniques continue to improve, and that many real-world problems such as military operations, conservation efforts, and video game entertainment will benefit significantly from progress towards more performant opponent modeling systems. Moreover, many other multi-agent application areas may benefit from advances in opponent modeling since these domains often still require modeling of another agent's hidden states. Conversely, we also argue that insights from other fields of research can spark innovation in opponent modeling.

2. Opponent Modeling

Research on opponent modeling has a long history, beginning in 1944 when John von Neumann and Oskar Morgenstern introduced the idea of game theory (Von Neumann & Morgenstern, 1944). Originally applicable only to zero-sum games, game theory concepts are now used to reason about a wide array of scenarios and have influenced the formation of a large number of related fields, including what we now recognize as opponent modeling. In practice, even the term 'opponent modeling' has been expanded beyond its denotation to include instances which are not strictly adversarial and in which some cooperation can be beneficial (Baarslag et al., 2016).

In this survey we define opponent modeling as the ability to use prior knowledge and observations in order to predict the behavior of one or more agents, whose internal states may not be fully observable, within the context of an adversarial game. Under this definition, both predicting a single opponent robot's position one time-step into the future, as well as predicting whether or not an opponent team's strategy will be, for example, aggressive or defensive in the second half, are forms of opponent modeling. Opponent modeling systems may be designed or trained such that they are prepared for *arbitrary* opponents, or opponents that operate under *certain assumptions*, such as rationality. In both cases, opponent modeling systems may be used against a specific opponent not known *a priori*, or be designed to be robust to any opponent in the assumed distribution. In the first case, an optimal strategy is the strategy that is best (under one's preferred definition of best) against the initially unknown or uncertain opponent strategy. In the second case, an optimal strategy is one that is best in expectation over all possible opponents. In this paper, these distinctions, when relevant, should be clear from context.

Opponent modeling approaches, regardless of the optimality criteria, or how abstract the predictions ultimately are, often make the following set of assumptions.

A.1 One of the following must hold: (1) There is no unique, optimal way to play the game, or (2) there is a unique optimal strategy, but it is unknown, or (3) there is a known, unique, optimal strategy, but the opponent is deviating from that strategy.

A.2 There is an observability partition over the game state $Y = Y_M \cup Y_H$, where the modelling agent cannot observe members of Y_H, but can observe members of Y_M.

A.3 Y_H may contain variables representing not only the immediate state of the environment, but also aspects of the opponent decision making process, such as plans, strategies, or intentions. Through interaction, observation and inference, the modeler can obtain information about the hidden variables Y_H.

A.4 The level of abstraction chosen by the modeler captures this information and facilitates a better estimation of the hidden state Y_H.

A.5 At the core of every opponent modeling approach is the expectation that the opponent strategy can be exploited and that more accurate estimates of Y_H will produce superior strategies.

We should note that this definition captures several informatically distinct opponent modeling problems. Perfect information games, such as Chess or Go, may restrict the

types of variables present in Y_H, but nonetheless fall under this definition. For example, Y_H may contain information about the tree pruning strategy in a tree-search Chess agent which could be exploited by setting up tactics from positions which are less likely to be expanded at search time. Partially observable games may contain private information but result in fully observable actions, such as in our robot soccer example where teams may select pre-determined plays in private, but both teams receive the same state information from a common vision system. Alternatively, many video games, such as Starcraft, have a fog of war, where many actions are also unobservable or partially observable. These types of games usually expand the variables in Y_H to contain more obviously strategic pieces of information, such as the composition or location of one's army.

Of particular importance in this list is A.5. This assumption is so universal that it is rarely stated explicitly. However, it is often tedious to measure directly, and unfortunately many studies do not report experimental results that answer this latent hypothesis. If opponent exploitation is the purpose of an opponent model, then overall system performance should be the experimental target rather than intermediate predictions or outputs of an opponent model. This is an important distinction since different methods of exploitation may themselves be exploited, and computing these methods often involves significant compute or memory resources which may not be available online — a limitation that earlier research on opponent modeling within adversarial search was keenly aware of (Donkers, 2003; Sturtevant, 2004). In fact, humans have been shown to generate strategies in the game rock, paper, scissors which allow exploitation, but which their opponent does not have the ability to recognize and exploit (Brockbank & Vul, 2021).

Furthermore, depending on the information captured by a given opponent model, the best-response in the long run, given the model's output, may not be to immediately exploit a perceived weakness. Understanding the value of the information provided by a model cannot be done by evaluating the model outputs in isolation, they must be evaluated *in situ*. Such setups confer many benefits. Although most papers we survey do not focus on the iterative design of strategic agents, opponent modeling under this definition may still play a role, aiding developers in uncovering different weaknesses or tendencies or even providing deeper strategic understanding of a game by highlighting hidden or emergent structure.

Clearly, these assumptions admit a range of possible solutions, which we explore in the following section. Some approaches are specific to certain games and some have potential to be generalized to a variety of applications. First however, we make a case for the necessity of opponent modeling approaches as opposed to more heavily model-based approaches, for games such as robot soccer. A natural first attempt to model adversaries in robot soccer would be to leverage the extensive work already done in game theory, where a large number of games have been studied. Indeed, optimal solutions can be computed for many classes of games (Carmel & Markovitch, 1996c; Bowling & Veloso, 2001; An & Sandholm, 2003), some of which are complex enough to model real-world phenomena (Shieh et al., 2012; Pita et al., 2008). Related work also includes approaches for opponents which can change strategy over time (Powers & Shoham, 2005) and 3-player games (Ganzfried et al., 2018). Recent work introduces several forms of counterfactual regret minimization (Farina et al., 2019; Brown & Sandholm, 2019; Davis et al., 2019) and deep reinforcement learning (Brown et al., 2020) to find Nash equilibria. However, it has been shown that computing optimal strategies when playing a limited lookahead opponent in an imperfect-information game is NP-hard

in all but the most restricted cases (Kroer & Sandholm, 2020). Moreover, many of these approaches require several pieces of information, such as sets of potential agent strategies, to be observable for all players, which is not the case in robot soccer. Furthermore, even those which do not rely on observable opponent strategies still require a known set of actions and rewards. In general, these sets may be unknown or difficult to compute.

One class of games, partially observable stochastic games (POSGs), can capture many important complexities (having multiple agents, partial observability, stochasticity, real-valued rewards, etc.). Although POSGs, which are closely related to decentralized partially observable Markov decision processes (Dec-POMDPs), can be solved exactly for small games (Hansen et al., 2004), in all but the most special cases exact POSG and POMDP approaches are intractable (Bernstein et al., 2002; Mescheder et al., 2011; Emery-Montemerlo et al., 2004; Ng et al., 2010; Egorov et al., 2016). Approximate POSG solutions exist (Ponsen et al., 2011; Ceren et al., 2021), but even approximate solutions are often still too computationally intensive. Often, finding an equilibrium strategy is not required to produce reasonable behavior. In this case, interactive POMDPs (I-POMDPs) have become popular, offering a way to model other agents' internal state. However, I-POMDPs are more complex than POMDPs, likely to have double exponential complexity even under some simplifying assumptions (Seuken & Zilberstein, 2008), and most work focuses on either state aggregation (Rathnasabapathy et al., 2006; Zeng & Doshi, 2012; Sonu & Doshi, 2015), or approximate solutions via Monte Carlo methods (Doshi & Gmytrasiewicz, 2009; Ng et al., 2012; Panella & Gmytrasiewicz, 2017) in order to make them tractable. In fully observable cases, nested MDPs may efficiently approximate I-POMDPs (Hoang & Low, 2013). Thus, the descriptive power of POSG-based and POMDP-based approaches is often too computationally demanding for many applications. Moreover, carefully developed and fine-tuned models of particular games rarely generalize or transfer well.

Another class of model-based approaches eschew comprehensive strategy understanding in favor of solving a simpler, more relevant problem. Work in plan recognition (Sukthankar et al., 2014) and goal recognition infers only the agent's current plan or goal. Because these techniques do not need a payoff matrix, they may be applied to tasks where joint action spaces are large, actions are taken asynchronously, or where payoffs cannot be calculated easily. When potential plans can be enumerated, a plan library can be used to match observations with hypothetical plans (Geib & Goldman, 2009; Kabanza et al., 2010; Zhuo & Li, 2011). However, this is not possible in many domains. To overcome the limitations of plan libraries, plan recognition can be reformulated as inverse planning (Baker et al., 2007), which allows the application of planners to discover potential agent plans (Ramırez & Geffner, 2009), distributions over plans (Ramırez & Geffner, 2010; Zhuo et al., 2012), and belief states within a POMDP agent (Ramırez & Geffner, 2011). These methods extend the applicability of plan recognition, but at the cost of significant compute resources.

A number of approaches have been proposed to improve plan recognition performance using ideas such as goal graph (Hong, 2001) analysis, pruning heuristics (Yolanda et al., 2015; Vered & Kaminka, 2017; Masters & Sardina, 2017), landmark (Hoffmann et al., 2004) detection (Pereira et al., 2017; Pozanco et al., 2018), and even meta reasoning about the time required to recognize symbolic plans and whether to take an action that could disambiguate plans (Fagundes et al., 2014). Unfortunately, these approaches still have several major drawbacks. First, they assume optimal agents, which violates the assumption that

optimal behavior is unknown, non-unique, or not being followed. Non-optimal agents have been studied (Keren et al., 2015; Tian et al., 2021a; Masters & Sardina, 2021), but so far this has not resulted in more effective algorithms for plan recognition. Second, all the methods mentioned above require an accurate model of the agent's internal decision making process, and many require noiseless observations. Third, plan and goal recognition algorithms alone do not allow generalization, since they are not designed to infer general properties of opponent behavior. These requirements prohibit the application of plan recognition to domains where either the internal state of the opponent is unknown, or for which an agent's decision making process is not modeled explicitly.

Recent advances in tracking technology and both supervised and unsupervised machine learning, coupled with the clear need to move beyond the present limitations of model-based approaches, have given rise to a growing number of techniques for opponent modeling. The remainder of this survey is dedicated to the classification and discussion of these techniques, as well as an overview of open problems and future research directions.

3. Overview of Existing Techniques

We find great diversity in the design and implementation details of opponent modeling systems even among approaches with similar applications. Although many papers focus most heavily on the mathematical models used for inference or predictions, in practice these models are not used in isolation. Rather, they are part of an entire data pipeline with potential constraints on both the input and the output. In fact, the class of inference algorithm is often the *dependent* variable in a system, constrained by the capabilities of the data pre-processing system, the quantity and quality of expert domain knowledge available, and the needs of the action selection module that uses opponent predictions. Our analysis has led us to identify three principal axes along which opponent modeling approaches vary.

1. *Method of Data Collection*: The method used to extract or pre-process data from raw initial input to be used by the learning algorithm. For example, features from a robot soccer match may be extracted by manually annotating footage of a game.

2. *Learning Algorithm*: The specific learning framework, such as support vector machines, decision trees, or neural networks, used to predict opponent state or behavior.

3. *Level of Abstraction*: The state space within which prediction and classification occurs. This may range from predicting the low-level actions of a single member of an opponent's team to classifying an entire team's collective behavior.

The ultimate effectiveness of an opponent modeling system is a complex function of all three variables. In the following subsections we describe the tradeoffs between different abstractions, learning algorithms, and data collection methods as they pertain to opponent modeling systems. Figure 1 situates behavior prediction algorithms in a generic opponent modeling pipeline. Here we should highlight the fact that the "best response" to an inferred opponent plan is not always to immediately exploit it. Not least because computing an exploiting response may take time, but also because showing the opponent that you have learned to exploit their strategy may sacrifice chances for greater exploitation later in the

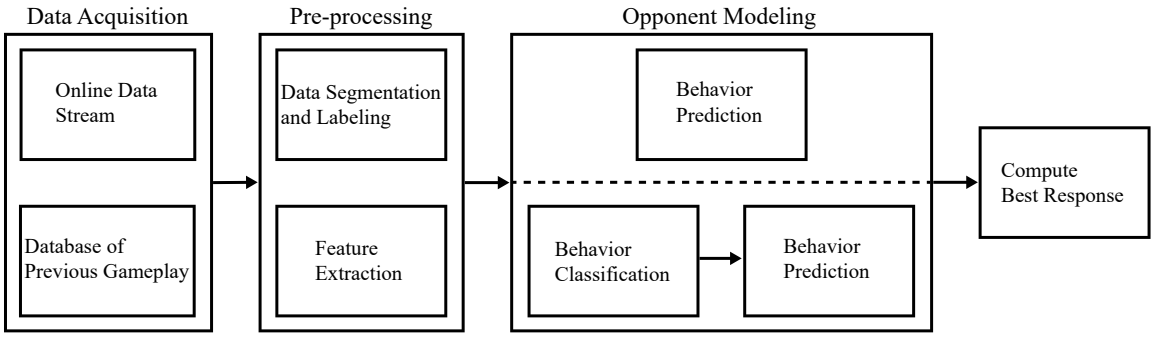

Figure 1: Overview of an opponent modeling pipeline. All opponent modeling systems begin with data gathered from previous gameplay, gathered during operation, or both. Most systems require a pre-processing step where data is augmented or transformed using feature extraction or segmentation and labeling techniques in order to properly format it for subsequent learning or inference. Some pipelines use both features and labels. Behavior prediction is either done directly or is preceded by a classification step, where opponent behavior is mapped to an abstract class and then the class identity is then used to predict behavior. Last, the predicted behavior is used to compute a response. Experiments measuring the efficacy of responses based on behavior prediction, rather than the accuracy of the prediction itself, are under-represented in the literature.

game. These chances could come through further refining the opponent model, or through an information asymmetry where the opponent is unaware of the modeling player's knowledge. The compute best response block in Figure 1 is intended as a general process representing any action taken in light of the new information presented by the opponent modeling system. That said, most papers surveyed here either assume best responses are played immediately, if available, or do not discuss potential best response strategies.

3.1 Data Processing and Domain Knowledge Requirements

By their nature, data driven approaches require some processing of raw data in order to fit a specified input format. The specifics usually rely heavily on both the game and the granularity at which the predictive component operates. For instance, predicting the position of a single robot will likely require trajectory information, while predicting an action such as shoot or pass might require more abstract features such as the given robot's position relative to its teammates. Comparing team-level strategies might rely on even higher level features, such as team centroid or the number of attackers and defenders. Furthermore, most models assume independence between behavior at the current time or game situation and behavior in the distant past or in a situation which is semantically different.

These assumptions are the result of applying expert domain knowledge and are valuable for their utility in simplifying the problem. However, the requirements these assumptions impose often create new, non-trivial data processing problems. For instance, if offensive and defensive behaviors are not correlated, then data which comes from an opponent's time spent on defense should not be considered when modeling their offense. Opponent modeling algorithms which operate on sequences of actions impose the requirement that

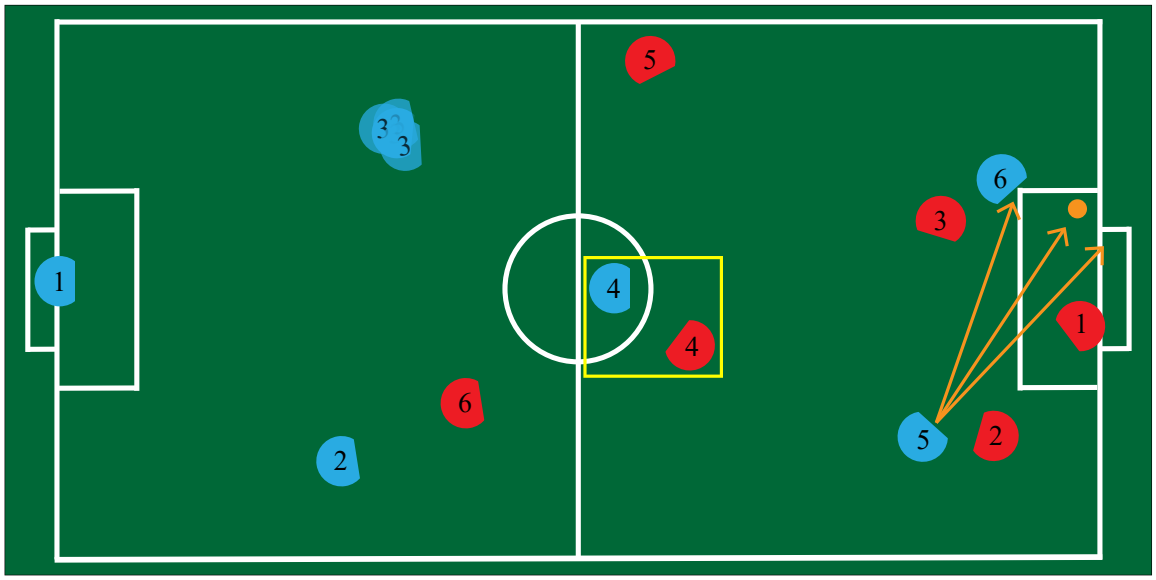

Figure 2: Collection of data processing challenges for opponent modeling systems. From left to right: Robot 3 in blue shows the uncertain, noisy nature of sensor input to most real-world opponent modeling systems. The pair of robots in the yellow box have an ambiguous relationship to the play happening to the right. Deciding whether they provide useful information is challenging. The orange arrows to the far right represent possible trajectories that robot 5 could have kicked, with the middle trajectory representing what was measured. Clearly, a pass or shot on goal was intended, but applying the correct label is difficult.

actions be correctly identified and extracted prior to being fed into a learning algorithm. This is not straightforward. Consider the case, shown on the right side of Figure 2, where a robot kicks the ball towards the goal, but has a teammate near the trajectory of the ball. Differentiating between a shot and a pass is critical to modeling the correct strategy, but can be difficult in practice. Similarly, many opponent modeling algorithms require labeling a subset of agents as involved or uninvolved in whatever is happening in the game. This is often ambiguous, as is the case with the number 4 robots highlighted in the center of Figure 2, and may require a sequence of inputs rather than a snapshot to accurately label. Moreover, observations in many real-world scenarios are noisy. This is illustrated to the left in Figure 2, where several samples from a distribution representing the likelihood of robot 3's position and orientation are shown. Thus, a variety of feature extraction and annotation systems have been developed in order to deal with some of the front-end problems created by the needs of the back-end algorithms.

Moreover, there are several different ways in which data can be affected by the collection process itself. These include whether data is gathered by conditioning labels or selection processes on certain observable properties (this is the most common), or whether we apply an intervention to the controlled behavior, thereby breaking the default causal structure to enforce a property of the game (Meek & Glymour, 1994). There are many other confounders in data collection, such as non-stationarity, where the opponent also learns or changes its

strategy over time, or other types of reasoning, such as deception or reasoning using the theory of mind. We touch on these topics in later sections.

In general, both automated feature selection and extraction as well as data segmentation are still unsolved problems which encompass large bodies of research in their own right. The majority of these works fall outside the scope of this survey. In light of the difficulty of this problem, many opponent modeling systems rely on human experts or manual annotation. Data is either hand-labeled with game actions or situations, or processed automatically by expert-designed scripts which use ad-hoc rules for extracting events and features. In these cases, the features used within the learning framework are fixed *a priori*. However, it is not clear that rule-based or hand-annotated feature extraction techniques provide optimal features for learning opponent models *at any level of abstraction*. Attempts to move beyond some of the limitations of manual annotation and ad-hoc feature extraction are often complicated by the fact that the ultimate design of an opponent modeling system is usually intimately related to, if not dependent on, the segmentation, feature selection, and dimensionality reduction tools available, and the complexity of these systems grows substantially as more coupled components are added.

As a result, a growing number of approaches have begun using unsupervised or clustering techniques for dimensionality reduction (Molineaux et al., 2009), feature selection using established techniques such as grafting and l_1 regularization (Vail & Veloso, 2008) or relational databases (Bhandari et al., 1997), and segmentation using various spatio-temporal models (Li & Chellappa, 2010; Takács et al., 2007; Beetz et al., 2005) or Gaussian mixture models (GMMS) (Perše et al., 2009; Kovalchik et al., 2020). GMMs in particular have produced some promising results in data exploration. For example, Kovalchik et al. (2020) estimate shot value in tennis via a simple generative model and a GMM trained on spatio-temporal data from the Australian Open. However, in general, comparisons between algorithms using unsupervised or semi-supervised feature extraction and hand-coded or rule-based systems using expert knowledge have not been performed at any notable scale.

Currently, there is a trade-off between knowledge exploitation and automation in rule-based data processing systems. The larger and more complex a set of knowledge becomes, the greater the descriptive power, but the less accessible it is to automatic processing techniques. This seems to be a fundamental characteristic of rule-based systems and some supervised approaches. Moreover, limitations in current automatic, unsupervised feature extraction technology may force the use of rule-based features, even though we lack any evidence to support the hypothesis that rule-based features are ultimately the best features. We hypothesize that unsupervised approaches to feature extraction such as clustering, representation learning, and manifold learning will not encounter this trade-off to the same extent. Therefore, it is plausible that the performance of opponent modeling systems would improve across the board given the ability to use complex feature sets which can only be extracted by unsupervised learning algorithms.

3.2 Choice of Learning Algorithm

A wide variety of frameworks have been applied to learning opponent models from data. Depending on the quality and amount of expert domain knowledge available, the size of the

state and action spaces, and the amount of coordination between opponent agents, modelers tend to employ one of three distinct approaches.

1. *Discriminative Role or Strategy Classification*: Behavior of one or more agents is classified according to a pre-existing set of classes. Class labels are then used to make a more nuanced prediction about future opponent actions.

2. *Goal-based Generative Models*: Behavior of one or more agents is classified based on a generative model of how agents achieve their goals. Class labels are then used to make a more nuanced prediction about future opponent actions.

3. *Policy Approximation*: A policy is learned for some state space and action space which approximates the true policy being executed by the opponent. Given a known state, the approximated policy predicts opponent actions or sequences of actions.

Discriminative models, generative models, and policy approximators have all been used with varying degrees of success. There is no dominant approach in terms of accuracy, since the choice of framework is typically constrained by the capabilities of the pre-processing systems and the availability of expert domain knowledge, both of which differ across applications. However, we do find a larger volume of discriminative models, perhaps because they are relatively easier to implement and evaluate. Here we present an overview of opponent modeling research categorized according to the learning or inference methods used.

3.2.1 Discriminative Role or Strategy Classification

Discriminative models estimate the probability of a variable x given an input y, $p(x|y)$. In discriminative opponent modeling, x is usually discrete, typically representing possible categories of behavior as defined by a domain expert. These categories range from individual agent roles such as attacker or defender (Vail et al., 2007; Sukthankar & Sycara, 2007; Beetz et al., 2006; Nair et al., 2004; Wendler & Bach, 2003; Li et al., 2009; Biswas et al., 2014; Devaney & Ram, 1998; Perše et al., 2009) to team-wide plays or strategy types (Laviers et al., 2009; Laviers & Sukthankar, 2011; Siddiquie et al., 2009; Fukushima et al., 2017; Kamrani et al., 2016; Riley & Veloso, 2000; Steffens, 2002, 2005; Schadd et al., 2007; Sadilek & Kautz, 2010; Lucey et al., 2013). One trait shared by all such discriminative approaches is that future opponent behavior is not directly estimated or predicted. Instead, a proxy for future behavior, the role or general strategy of one or more agents, is predicted. This classification is then used by other processes to produce a distribution of possible future opponent actions and game states.

Support Vector Machines A nice example of this pipeline is the work of Laviers and Sukthankar (2011) and Laviers et al. (2009), where defensive play types (team-wide strategies) in simulated American Football, such as blitz or coverage, are identified using a support vector machine (SVM); then, a plan to counter the opponent's strategy is constructed. Spronck et al. (2010) also use SVMs to classify players' preferences for different game objectives in the game Civilization from sequences of actions, and Fukushima et al. (2017) use SVMs, along with neural networks and random forests, to create an ensemble of learners which use position information to identify free kick strategies in RoboCup simulations. Siddiquie et al. (2009) and Li et al. (2009) both use hand-annotated footage of real American

football games to train play recognition systems. The former group uses SVMs, while the latter proposes a discriminative temporal interaction manifold (DTIM) over which multi-modal probability distributions corresponding to different play types are learned. The DTIM has some desirable properties including being view-stable (although not view-invariant) and not relying on expert knowledge, except for labeling training data.

Case-based Reasoning Case-based reasoning (CBR) is also a popular technique due to its ease of implementation and the ability of domain experts to supply examples directly, rather than having to specify rules which describe the behavior of interest. Steffens (2002, 2005) and Wendler and Bach (2003) both employ CBR to identify team strategies in RoboCup using expert-designed metric features and custom similarity functions. Ahmadi et al. (2003) present a two-layer CBR system in simulated soccer, where the goal is to predict the position and velocity of all other players at an unknown time in the future, when possession changes. They accomplish this by comparing sequences of metric features, including player positions and velocities relative to the ball. In the bottom layer, they use these features in robot-centric coordinate systems, and in the top layer they use the same features transformed into a global coordinate system.

Beyond robot soccer, CBR is also popular in video game AI research. Fagan and Cunningham (2003) use CBR along with a plan library to predict plans in the game Space Invaders, and Hsieh and Sun (2008) learn to predict opponent strategies and unit compositions in the real-time strategy (RTS) game StarCraft based on the timing of buildings. Similarly, Weber and Mateas (2009) use the times at which the first technologies, buildings, and units are researched, built, and trained along with a form of CBR called non-nested generalized exemplars (Martin, 1995) to predict StarCraft strategies from an expert-generated set. They also use M5 (Quinlan, 1993) to predict the time the first unit will appear. Farouk et al. (2017) use CBR to predict the opponent's strategy in another RTS game, GLest. One drawback of CBR is that it is exceptionally difficult to evaluate its efficacy relative to other approaches. Because the performance of CBR systems relies so heavily on custom functions and features, the question of whether errors are due to imperfect expert-designed components or shortcomings of the CBR framework in general are rarely answerable.

A number of other approaches to classifying team strategy have also been investigated. Schadd et al. (2007) propose hierarchical fuzzy classification for determining player strategies in RTS games based on their actions, such as 'build' or 'attack'. Sadilek and Kautz (2010) use Markov logic networks to identify group events in a real-life capture the flag game where player positions are given by GPS. A compelling example of incorporating information-theoretic concepts into feature selection is presented by Lucey et al. (2013), where Linear Discriminant Analysis is used to identify English Premier League teams based on spatiotemporal entropy data from manual annotation.

Expert Systems and Metrics The application of domain knowledge to discriminative modeling is rarely limited to the definition of discriminator outputs. Expert-designed metric features and sequences of actions, whose discretization and definition themselves require domain knowledge, are common. Riley and Veloso (2000) use sequences of actions and decision trees to classify RoboCup strategies according to a predefined set, and Kaminka et al. (2002) construct tries based on sequences of actions extracted via expert-defined rule sets. The dependency detection algorithm (Howe & Cohen, 1995) is then used to

predict online which previously observed team is playing in simulated RoboCup. Bombini et al. (2010) also generate sequences of actions via expert-defined rules. An extension of the apriori algorithm (Esposito et al., 2008) is run on these sequences to find features, and construct Boolean feature vectors from these sequences. A k-nearest neighbor algorithm using the Tanimoto measure (Duda et al., 2012) predicts the current opponent from a set of previously seen teams in simulated RoboCup. Expert-defined features are not unique to simulated RoboCup. For instance, Kamrani et al. (2016) apply both sequences of actions and expert-designed metric features within decision trees and neural networks in order to classify military tactics in simulation.

Attempts to classify single agent behavior within a multi-agent game take largely the same approach as their team-wide counterparts. Vail et al. (2007) use sequences of expert-designed metric features along with conditional random fields to identify which agent is the seeker in a game of tag, and Sukthankar and Sycara (2007) use both SVMs and an application of Dempster-Shafer theory over expert-designed templates to classify player roles in a group combat game. Some approaches such as Isaac, introduced by Nair et al. (2004), construct a hierarchical model, solving multiple classification problems simultaneously under the same framework. Isaac uses decision trees and probabilistic finite automata to examine traces of opponent actions and metric features, and then classifies behavior at a variety of levels including both individual agents and entire teams. Ledezma et al. (2009) also solve multiple problems simultaneously with their Ombo system. Based on data from RoboCup simulations, they construct both decision trees and regression trees to predict future actions, such as kick, or turn, along with their real-valued parameters, such as velocity, or angle. Ombo uses PART (Frank & Witten, 1998) to label actions and build the decision tree and M5 to label action parameters and construct the regression tree.

Hand-built discriminative models for single agent classification are also common. Perše et al. (2009) identify roles by comparing action sequences to manually defined templates via the Levenstein distance. Biswas et al. (2014) apply a simple and effective model to RoboCup, where a subset of raw features are matched to behavior templates via an ad-hoc distance measure to determine an agent's role, which is then used to formulate a coercive attack plan. A thorough presentation of the challenges and assumptions made in opponent modeling systems is given by Devaney and Ram (1998), who use sequences of positions from US Army training exercises to classify pair-wise relations between soldiers based on expert-designed rules. Sequences of relations are then matched to templates to infer higher level activity. Importantly, this algorithm does not need to consider the whole team.

At the boundary of opponent modeling and feature extraction, work by Beetz et al. (2006) in which expert-defined templates and decision trees are used to label agent actions, highlights the importance and challenges of modeling failed actions. Action sequences are attractive inputs to opponent modeling systems since they seem to transform a continuous, high-dimensional problem into a discrete, lower-dimensional one. However, verifying the completeness of an action set such that all observations are explainable by some sequence of actions and acquiring accurate action labels are still unsolved problems.

Game Theoretic Approaches At the other end of the spectrum, research with a clear lineage back to game theory continues. Wang et al. (2011) propose a game-theoretic approach that adaptively balances exploitability and risk reduction. The opponent's next

action is modeled as a set of possible actions that contain the actual action with a high probability. In their application to table-tennis, the actions correspond to one of three serve locations and the response actions are thus setup locations for the return. The algorithm is 'safe' as the expected payoff is above the minimax payoff with a high probability, and can exploit the opponent's preferences when sufficient observations have been obtained. In multi-agent settings, Shen and How (2019b) have used reinforcement learning over belief-space in Bayesian games to model and respond to multiple potential opponent strategies, or types. The mapping from belief to action is ultimately a neural network, and the approach is evaluated on a small grid world domain.

Summary Discriminative approaches are usually intuitive, easy to use, and easy to understand. However, they also have several serious drawbacks in the context of opponent modeling. First, the labels generated by these models rarely have meaning in the raw game state space. Instead, as mentioned before, the labels are proxies for behavior patterns, and there must be a subsequent process which translates an expert-designed, semantic label into an actionable distribution over future game states. This process is difficult and often ad-hoc, limiting the power of discriminative opponent models.

Second, there is a strong assumption underlying discriminative opponent models: the opponent is choosing from a range of strategies or behaviors which closely match the labels chosen by the modeler. This is clearly not true in general, and poor agreement between classifier taxonomy and the opponent's internal representation will decrease performance of downstream tasks. Third, discriminative models rely heavily on expert knowledge. From input features to class labels, many discriminative approaches are only as good as the intuition given to them by their programmers. Automatically generating labels via unsupervised techniques addresses both of these shortcomings, and we view progress on that front as vital to the progress of discriminative opponent modeling as a whole.

3.2.2 Goal-based Generative Models

Generative models estimate to joint probability of hidden variable x and observation y, $p(x, y)$. In practice, generative models and discriminative models often perform the same function in opponent modeling frameworks. They assign a semantic label to the behavior of an agent or group of agents, which is then used as a proxy for calculating the probability of some future state. Generative models allow incorporation of even greater amounts of expert knowledge, since the relationship between x and y must be explicitly given.

Bayesian Networks A classic example is the work by Intille and Bobick (1999, 2001), who designed Bayesian networks to predict American Football play types from sequences of player positions and expert-designed features. Bayesian networks have also been used by Ball and Wyeth (2003) to classify agent behaviour in robot soccer, and by Riley and Veloso (2001) who use naive Bayes over sequences of robot positions in addition to a set of pre-defined models to predict future positions in simulated robot soccer set plays. They also provide one of the better explicit descriptions of what their underlying model is actually computing, including explicitly stating the static opponent strategy assumption. Over a decade later, Bayesian networks remain a popular framework for creating generative opponent models, as in Wei at al. (2013), who are able to write down a complete parameterization of a tennis rally, enabling them to predict the outcome of various strategies (eg. types, lo-

cations, and velocities of shots). Recently, Torkaman and Safabakhsh (2019) use Bayesian networks to predict build orders in StarCraft by training them on timestamped game logs. These features are more general than those used by Synnaeve and Bessiere (2011), who employ EM to label logs for the presence of expert-designed features corresponding to different StarCraft strategies, and then use Bayesian networks to predict strategies online from a predefined set. Some research using generative models has also been concerned with modeling the probability that the opponent is being deceptive. In work by Stankiewicz and Schadd (2009), Bayesian inference over the local board dynamics and statistics from past matches are used to predict the rank of a given piece (a hidden variable) in the game Stratego. Essential to this process is determining whether the opponent is bluffing, for instance, by posturing aggressively with a relatively weak piece.

Hidden Markov Models Hidden Markov models (HMMs), a type of Bayesian network, are also popular generative models for opponent modeling. Intuitively, they align with our understanding of how many systems, both autonomous and human-dependent, are designed. A common architecture in robot soccer, for instance, is the skills, tactics, plays paradigm (Browning et al., 2005), wherein robot behavior is more or less governed by a hierarchy of internal state machines, whose hidden states are both the object of interest in opponent modeling and also concisely described by models like the HMM. The idea of a strategy as a set of hierarchical policies is explained well in work by Bui et al. (2002), who model such a strategic agent with an abstract hidden Markov model. Sequences of actions and positions are used to predict which policy, from a predefined set, the agent is following. Saria and Mahadevan (2004) extend this work to consider multiple agents. A goal-based version of the above is presented by Willmott et al. (2001), who use hierarchical task networks to predict opponent goals and subgoals in the game of Go. Blaylock and Allen (2006) introduce the cascading HMM, another form of hierarchical HMM designed for goal inference.

In addition to being organized hierarchically, HMMs can be used in ensembles to estimate the likelihood of a large number of different internal states simultaneously. Han and Veloso (2000) use an ensemble of expert-designed HMMs, a subset of which are selected at each time-step online, to classify the role of soccer robots using sequences of robot positions. Similarly, Sukthankar and Sycara (2006) identify predefined military tactics using a combination of template matching via RANSAC (Fischler & Bolles, 1981) and HMMs.

Simulation Some generative approaches do not use exact inference models, relying instead on black-box simulation or expert knowledge. Kuhlmann et al. (2006) use linear regression and knowledge about an automated agent's formation control in simulated soccer to solve for parameters of the opponent's formation given a stream of position data. Butler and Demiris (2009) compare raw observations to simulation results from a hand-designed generative model. Using a domain-specific distance function, they infer the internal state of agents in a military simulation (attack, defend, etc.), as well as the group formation. Floyd et al. (2017) also use simulations to bootstrap a case-based reasoning system for identifying and classifying enemy combatants engaged in beyond-visual-range air combat. In some cases, generative models can become very elaborate, as with Shen and How (2019a), where Markov decision processes are specified for both adversarial and neutral agents, along with rationality and deception parameters. A neural network is then trained using a variant of Q-learning which maximizes entropy (minimizing exploitability). The goal is to predict

whether an agent is neutral or adversarial in a simulated checkpoint scenario. Wang et al. (2017) study opponent modeling in Phantom Go, a partially observable version of Go. They propose an algorithm to predict the opponent's next move which uses sequences of game states as input to a simulation of the opponent's own Monte Carlo tree search through belief space, using quantal response to calculate beliefs.

Summary Generative models do not enjoy the widespread popularity in opponent modeling of their discriminative counterparts. We hypothesize that this is due to the increased burden of defining the generative process of observable variables. Additionally, many of the benefits of generative models may be impossible to realize in opponent modeling domains where knowledge about the opponent is too scarce to build a complete model. Moreover, opponents can inadvertently exploit the construction of a particular generative model simply by using a different reasoning process than the one modeled.

3.2.3 Policy Approximation

Policy approximation is a special case of function approximation, where the function being approximated is a policy $\pi : S \to A$. S is a set of states and A is a set of actions. In some cases $\pi(s)$ may map to a distribution over actions instead of a single action. Thus, a policy approximator seeks the probability that action a is selected given state s, $p(a|s)$. In practice, policies may be represented in a variety of ways, such as state-action tables, decision trees, or neural networks, and they may operate on state definitions containing discrete variables, continuous variables, or a combination. We do not restrict our attention to any particular policy or state representations. In contrast to both discriminative and generative modeling, policy approximation directly produces the probability of future game states, since action probabilities and their impact on the game state can be used without any intermediate step. Expert domain knowledge can still be incorporated into policy approximation frameworks in the form of the state and action sets S and A.

The primary weakness of the policy approximation approach is the poor tradeoff between state-action expressivity and the amount of data and compute resources required to achieve good approximations. Experiencing every state-action pair is often infeasible for continuous or even large discrete spaces, and the computation time for many popular methods scales poorly with the size of the state space. Thus, policy approximation methods for opponent modeling all deal with these limitations in some way, most often by abstracting the state space to a reasonable size or by using less expensive and less accurate approximators, such as those which produce a single action instead of a distribution over actions.

Decision Trees One example of the latter approach is a hand-built behaviour model by Stone et al. (2000) which conservatively estimates opponent capabilities and the probability that the opponent could reliably stop a given plan from achieving its goal. In addition to discriminative classification, decision trees can also be used as policy approximators, as in work by Visser and Weland (2002) who use a sequence of metric features to predict the opponent's next action in robot soccer. Markovitch and Reger (2005) introduce the concept of policy "weakness". Weakness is the deviation of an opponent's policy from that given by an oracle, which may be a human expert or a more computationally intensive version of the policy generation algorithm. Given a sequence of actions, they use a decision tree to predict the opponent's weakness at a given state online. More recently, Kar et al. (2017) have

used ensembles of decision trees to predict poaching activity in Queen Elizabeth National Park in Uganda. Although simple, in practice these models often outperform their more complicated, game-theoretic counterparts. Previously, Yang et al. (2014) modeled poaching as a Stackelberg security game using subjective utility quantal response (Nguyen et al., 2013). An extension of this work, by Rahman and Oh (2018) modulates rewards with simulated annealing to perplex opponents.

Game Theoretic Approaches Many other modifications to game-theoretic approaches to incorporate data online have also been explored. Chakraborty et al. (2013) developed Tommba, an algorithm which, given an informative feature set which is at least as compact as the action history and a set of strategies which contains the opponent's true strategy, can guarantee a certain level of performance against memory-bounded agents. The need for these assumptions is a result of work by Ganzfried and Sandholm (2011), who show that a variation of the No Free Lunch Theorem applies to adversarial games. It is not possible to exploit an opponent's strategy, by deviating from the equilibrium strategy, without creating the possibility of being exploited oneself. Although exceptions were later discovered (Ganzfried & Sandholm, 2015), this general tradeoff between exploitation and safety is sometimes called the "get-taught-and-exploited problem" (Sandholm, 2007), and has been applied in spirit to non-game-theoretic domains as well (Biswas et al., 2014). The concept of agent misdirection has also spawned interest in strategies for counter-misdirection (Chen & Arkin, 2021). In other cases, opponent modelling has been shown to have neutral or even negative effects on an agent's expected reward (Zhang et al., 2020). Understanding the tradeoffs between exploitability and exploitation is still an active area of research, and in some cases, such as Johanson and Bowling (2009), researchers have found ways to use data from past games in order to bias strategies away from their equilibria in a manner that can be beneficial in the short term. Other work has sought bound performance against unknown opponents (McCracken & Bowling, 2004) or bound the exploitability of one's own exploitative strategies (Johanson et al., 2008).

Deterministic Automata Another method for simplifying the inference problem is to model opponent strategies as deterministic finite automata (DFA). Although finding the minimal consistent DFA is NP-Complete (Gold, 1978), and even bounded approximation is hard (Pitt & Warmuth, 1993), finding unbounded approximate solutions that work well in practice is tractable. Furthermore, as in the case of HMM-based opponent models, representing systems as DFAs has a certain intuitive appeal. Fard et al. (2007) use a combination of expert-designed and data-mined payoff matrices to inform action selection in RoboCup Simulation. Opponent strategy is represented as a DFA, and the problem is made tractable by identifying cycles within opponent behavior. Rovatsos et al. (2003) use the $US\text{-}L^*$ algorithm (Carmel & Markovitch, 1996b) to construct DFAs representing opponent strategies from sequences of actions, and Marin et al. (2005) later extend this work for use in RoboCup simulation by allowing simultaneous encounters with multiple agents and adding environmental information, such as the score. Iglesias et al. (2009) try to detect the deviation of opponent strategies from some base strategy, given examples of both. They store sequences of actions in a trie, and then prune the trie based on the frequency with which certain branches are visited. An ad-hoc similarity function is designed to determine when a detected sequence of actions matches a pattern stored in the trie.

Abstract Markov Models In contrast to the approaches above, many opponent modeling systems do not bypass computational limitations by sacrificing inference accuracy. Since exact inference in raw observation space is often intractable, these methods use abstractions over the observation space to create a state space of manageable size. The most straightforward example is work by Baez (2015), where sequences of opponent positions are used to learn transition probabilities in a Markov chain, the states of which discretize a robot soccer field. Similarly, Kim and Kim (2017) use an action table compiled from opponent action data to seed a Monte Carlo tree search in a fighting video game. More expressive models, such as semi-Markov decision processes, are employed by Yin et al. (2016) which, together with Rao-Blackwellized particle filtering (Doucet et al., 2000), allow them to use sequences of positions in order to predict the destinations of agents in RTS games.

POMDP Variants Wunder et al. (2011) introduce the parameterized interactive POMDP (PI-POMDP) in their study of opponent strategy in the Lemonade Stand Game. PI-POMDPs, and I-POMDPs in general, explicitly model interactions with other agents in the state space, given some opponent policy. The PI-POMDP allows the interactions between agents to be modeled considering multiple opponent policies and is shown to uncover inherent 'levels' of reasoning, with higher level strategies correlating with more successful agents. In a decentralized setting, Hoang et al. (2017) show how macro-action decentralized POMDPs (MacDec-POMDP) can be used to learn distributions of opponent strategies and formulate policies which perform well against those strategies. The strength of this approach is the macro-action, which allows efficient abstraction and simplifies the bookkeeping in modeling decentralized systems. It also limits this approach's applicability, since it requires expert formulation of macro actions. In practice, POMDP-based approaches require significant abstraction in order to work on real-world systems.

Deep Neural Networks Neural networks have also been used in a variety of ways to model opponent policies. Lockett et al. (2007) use genetic algorithms to evolve a population of neural networks via SharpNEAT (Green, 2004), a variation of NEAT (Stanley & Miikkulainen, 2002), which learns to play against opponents in the card game GuessIt (Isaacs, 1955). Opponents are represented as a linear combination of four cardinal strategies, and the system uses these strategies along with the action history of the opponent to produce a probability distribution over the next action. Tang et al. (2020) train a neural network incrementally online to predict the next action of agents in the fighting game FightingICE (Lu et al., 2013). Recently, deep Q-learning models have been proposed for opponent modeling. He et al. (2016) formulate a multitasking Q-learning problem where a representation of the opponent policy is learned alongside a policy for playing against the opponent. The resultant policy is tested on a simple soccer simulation. Hong et al. (2017) present similar work. In StarCraft, Synnaeve et al. (2018) use an autoencoder with a recurrent neural network (RNN) encoder built from convolutional long short term memory (LSTM) cells to predict unit locations in a downsampled grid and unit types at a fixed but arbitrary time in future, given full history of partial observations, terrain features, and team factions. Deep learning offers an attractive alternative to model-building, but on physical systems it is often challenging to get enough data for deep networks to learn effectively. The tradeoff between the generalizability of deep learning and the interpretability and robustness of model-building and model-based systems still presents a challenge, even to experts. Of late, there have

been many works using deep learning which learn best-response strategies directly, such as AlphGo (Silver et al., 2016), and thus contain information about opponent strategies latent within some internal representation. Here, we focus on explicit opponent modeling systems and thus omit review of systems which indirectly model opponent behavior.

In an attempt to address situations where opponents may change strategies or policies over time, termed non-stationarity, Chen et al. (2020) train a neural network to reproduce an opponent's state-action sequences from a discrete soccer simulator, but add the entropy of the predicted policy as a regularization term in an effort to avoid over-fitting to the current policy. Everett and Roberts (2018) also use neural networks to model opponents based on observations of past actions. In a real-valued adversarial orchard simulator, they propose dealing with non-stationarity by swapping between multiple opponent models online, where swaps are triggered by accumulated prediction error as well as model uncertainty, which is modeled using Monte Carlo dropout. Zheng et al. (2018) present perhaps the most compelling framework for modeling non-stationarity to-date. Under the framework of Bayesian policy reuse (Rosman et al., 2016), they model opponent policies via neural networks, and use policy distillation (Rusu et al., 2015) to bootstrap or hot start network learning online when a novel opponent policy is encountered.

Summary In general, policy approximation approaches seem promising due to their ability to estimate future states directly. However, these approaches have an important weakness. Policy approximation directly on the raw game state quickly becomes intractable as there may be too many unique game states which require observations in order to form quality approximations. On the other hand, policy approximation on very compact, abstract states risks washing out important nuances in agent behavior. Policy approximations may also overfit to a small number of examples, and if the environment is stochastic, the policy approximator may have trouble distinguishing between likely outcomes and fluke results.

3.2.4 Additional Methods

There are also methods that do not explicitly model opponent behavior or use supervision. These methods are useful for discovering knowledge about types of strategies different agents may take (Trevizan & Veloso, 2010; Cliff et al., 2013, 2017; Xu & Julius, 2018), or knowledge about different patterns which may be present in the raw observation space (Erdogan & Veloso, 2011; Yasui et al., 2013; Adachi et al., 2016). They are often valuable preliminary steps in the development of opponent modeling systems, offering additional insight into the domain which can be combined with expert knowledge.

Clustering Methods One common technique is to use unsupervised clustering to discover informative features and classes. For example, Adachi et al. (2016) and Yasui et al. (2013) use hierarchical agglomerative clustering (HAC) (Müllner, 2011) to cluster free kick strategies based on deployment locations and sequences of actions, respectively. One particularly compelling approach by Erdogan and Veloso (2011) uses HAC to cluster robot trajectories, allowing prediction of the eventual goal locations from partial trajectory information online during RoboCup. This is a good example of using unsupervised learning to discover class labels. Similar in spirit, Lattner et al. (2005) cluster sequences of expert-defined actions and metric features using custom distance functions, and then use these templates to predict subsequent actions in RoboCup simulation. Beyond RoboCup, Hayes

and Beling (2018) use HAC to cluster build orders in StarCraft. Their analysis shows it is even possible to distinguish individual players based on the clusters they produce and the evolution of clusters over time, as different strategies become more or less common during the course of the game. Alternatively, Xu and Julius (2018) take a top-down style approach, wherein signal temporal logic is used to express multiple levels of behavior patterns using player position data from real soccer matches.

Information Theoretic Methods There are also many non-clustering approaches that can be equally enlightening. Cliff et al. (2013, 2017) analyze sequences of positions of different sets of agents from RoboCup Simulation to determine which actions are correlated across agents using Shannon information dynamics. Among the advantages of information-theoretic approaches is that patterns detected from one opponent may be indicative of patterns in the game in general and thus transferable to contests with new opponents. Furthermore, they may validate expert-designed features, indicating whether or not they are truly informative. A nice example of these benefits is seen in work by Trevisan and Veloso (2010), where principal component analysis is used to generate a number of features. Features, which also include the current strategy of the controlled team, are stored as columns of a matrix E, and when the Frobenius norm of the expression $E^A - E^B W$ is minimized, where W is a vector with non-negative entries, the norm is proportional to ability of episode B to explain episode A. This allows the similarity between strategies to be quantified, and in their study the method is applied to defensive strategies in RoboCup. Recently, Grover et al. (2018) propose analyzing the embedded representations of RoboSumo (Al-Shedivat et al., 2017) agents, trained via an autoencoder, in order to predict the outcomes of different matchups. Similarly, Papoudakis and Albrecht (2020) use a variational autoencoder to learn embedded representations of agents in various problems in the multi-agent particle environment (Mordatch & Abbeel, 2018) and analyze these representations, in part, by calculating their mutual information neural estimation (Belghazi et al., 2018).

3.3 Model Abstraction Level

Almost all approaches presented here abstract the observed game state to some smaller state space. For example, abstracting each agent's raw position and velocity observations to a role, such as attacker or defender, via classification. These abstractions can be thought of as functions which perform a lossy compression and change of basis akin to principal component analysis or its non-linear counterparts, and thus form a new basis upon which approximations of dynamics in the original space may be calculated more efficiently.

Abstraction choices within opponent modeling systems are motivated primarily by expert domain knowledge, which is consistent with the model design process in many other fields. However, popularity does not entail optimality. As has been observed in many machine learning applications, *the effectiveness of a particular abstraction is correlated with the degree to which the space of game states can be approximated by a manifold defined over a set of basis vectors (feature set) produced by the given abstraction*, where our game state Y is again the union of observable state factors and hidden state factors of all agents. That is, if a feature set does not contain enough information to distinguish between strategically distinct game states, then learning using these features will not create effective models.

Abstractions are related to two of the most fundamental questions in opponent modeling. First, the hypothesis that expert-designed abstractions produce superior approximations to learned abstractions has not been decided conclusively. Second, it is not clear which level of granularity is optimal for modeling opponent behavior in robot soccer, or any other similar game. The question of exactly what opponent modeling systems ought to predict is essential. Unfortunately, it is also a difficult and tedious variable to experiment with, since redefining the space of predictions typically means redefining most other components of the system. Furthermore, these questions present challenges in comparing different opponent modeling approaches and understanding how different abstraction choices affect performance.

One way to frame the different abstraction choices made by modelers is to recognize whether predictions are made with respect to a single agent, a subset of agents, or all agents. For example, in RoboCup one might estimate an individual agent's role, such as attacker or defender, in order to predict the subset of the game state with respect to that agent alone. One might also estimate the likelihood that one robot intends to execute a pass to another robot. In this case, the subset of the game state for both robots is predicted as well as the state of the ball. Further, one might estimate the likelihood of a specific play from a given set, which could produce better predictions for all game state variables under the opponent's control. These levels of increasing abstraction form a natural hierarchy, and some existing systems already presented here try to exploit this structure. Other approaches have identified similar abstraction hierarchies for single agents executing plans, which are composed of actions, which in many domains are parameterized by real-valued inputs.

Although there is a clear hierarchical structure in many multi-agent activities, the practice of modeling adversaries hierarchically is not common. Moreover, there does not seem to be a uniform strategy for implementing such hierarchies. Some approaches use only the highest and lowest levels of information, but nothing in between. Others use multiple low- and mid-level models, but do not combine them to build group-wide estimators. In order to facilitate comparisons between, and discussion about, opponent modeling systems, we present a three-class taxonomy of state abstractions for multi-agent opponent modeling.

Definition 1. Given a set of opponent agents O, an **atomic abstraction** computes an abstraction function $g : \{Y_A^{t-k}, \ldots, Y_A^t\} \rightarrow Z$. Here, $Y_A^t \subset Y$ for all t, and $\forall y \in Y_A$, $\frac{\partial \Gamma(o_i)}{\partial y} \neq 0$, where $\Gamma(o_i)$ represents the capacity for agent o_i to affect state factors.

Example 1. Use only the location of an agent to classify it as an attacker or defender.

Definition 2. Given a set of opponent agents O, a **branch abstraction** computes an abstraction function $g : \{Y_B^{t-k}, \ldots, Y_B^t\} \rightarrow Z$. Here, $Y_B^t \subset Y$ for all t, and $\forall y \in Y_B$, $\exists i$ such that $\frac{\partial \Gamma(o_i)}{\partial y} \neq 0$, where $\Gamma(o_i)$ represents the capacity for agent o_i to affect state factors.

Example 2. Use the velocity of multiple agents to classify them as converging or diverging.

Definition 3. Given a set of opponent agents O, a **complete abstraction** computes an abstraction function $g : \{Y_C^{t-k}, \ldots, Y_C^t\} \rightarrow Z$. Here, $Y_C^t \subseteq Y$ for all t, and $\forall y \in Y_C$, $\exists i$ such that $\frac{\partial \Gamma(o_i)}{\partial y} \neq 0$, where $\Gamma(o_i)$ represents the capacity for agent o_i to affect state factors. Complete abstractions must also be on-to: $\forall o_i \in O$, $\exists y \in Y_C$ such that $\frac{\partial \Gamma(o_i)}{\partial y} \neq 0$.

Example 3. Use the average team location to classify a formation as aggressive or defensive.

Although these definitions use partial differentiation notation most common with continuous variables, any game state variable, continuous or discrete, is covered under this definition. Given a discrete variable, we simply use a finite difference rather than a limit expression when computing the partial derivative, as is common practice in fields like computer vision. These three classes represent the primary methods for applying change of basis in opponent modeling systems from game state variables to a smaller set of variables, which often carry semantic meaning to the modeler. This notation makes explicit some important distinctions. Opponent modeling systems may use information about the environment that is not directly controllable by the opponent. The causal source of this information is distinct from direct observation of opponent behavior and the abstract features generated from these observations. This same phenomenon can occur between different agents in a multi-agent setting and communicating this clearly is important to understanding why different techniques may or may not be suitable for different domains.

The interactions and tradeoffs between different abstraction levels are still not well understood outside of anecdotal evidence, and quality comparisons between methods are rare. Specifically, opponent modeling research would benefit from comparisons between methods applied to the same domain, but which use different abstraction classes. Similarly, research investigating how abstractions of the same class perform using different types of features could also significantly advance our understanding of the reliability and generality of many proposed techniques.

We surmise that, ultimately, hierarchical approaches will be most effective, wherein specialized classifiers or policy approximators are used for different parts of the hierarchy. Depending on the exact architecture this may fall under the definition of layered learning (Stone & Veloso, 2000). Furthermore, as access to data, computing resources, and effective opponent modeling software becomes easier, we envision hierarchical models becoming more tractable. The levels of abstraction defined here map naturally to a hierarchical structure, and we hypothesize that organizing hierarchical opponent modeling approaches after this structure will produce stronger results.

3.3.1 Approaches for Learning and Their Relation to Abstraction

If X is the space of raw observations, such as images from a video feed, Y is the game state space, and Z is the space of an abstracted representation of the game, such as player roles, most opponent modeling systems can be expressed as composite functions of the form

$$g^{-1}(h(g(f(X)))), \tag{1}$$

where $f : X \to Y$, $g : Y \to Z$, and $h : Z \to Z$. f extracts game state features from raw sensor data, and depending on the game and whether the system is operating in the real world, may be the identity function. $g()$ takes the raw game state and constructs a compressed or abstracted version, the definition of which is chosen by the modeler. $h()$ performs predictions in the space of Z. The study and development of the functions $f()$ are largely outside the scope of this section, and we assume they are sufficient although imperfect. In discriminative and generative approaches, $g()$ and $h()$ are separate functions, while in policy approximation systems, the policy approximator learns $p(\cdot) = g^{-1}(h(g(\cdot)))$. To be clear, in this notation, $g^{-1}()$ is not the inverse of $g()$ in a strict mathematical sense.

In fact, $g()$ may not be invertible. We write $g^{-1}()$ to denote a function where dom $g^{-1}() =$ codom $g()$ and codom $g^{-1}() =$ dom $g()$.

In practice, both determination of Z and design of $h()$ are highly non-trivial, and we return to a tension we have seen before: the tradeoff between the descriptive power and completeness of a set of knowledge, and the technical and implementation costs of exploiting that knowledge maximally. Learning robust functions for $g()$ is usually far easier than approximating the policy and thereby learning $p(\cdot) = g^{-1}(h(g(\cdot)))$. However, learning an approximate policy $p()$ removes the need to engineer Z and $h()$ altogether. This is a familiar story in machine learning, as neural networks have become increasingly good at learning their own features, eliminating the need to hand engineer feature spaces such as Z. Unfortunately, relatively few works presented in this survey discuss their choice of $h()$ explicitly, and many do not validate their approaches with experiments that test the entire pipeline, leaving the question of whether or not the proposed methods actually create better opportunities to exploit opponent strategies unanswered.

4. A Foundation for Understanding Game State Trajectories

Over the course of any adversarial engagement, the true game state $y^* \in Y$ will take on a number of values sequentially, $\{y_0^*, y_1^*, \ldots, y_n^*\}$, creating a trajectory through the game state space. Eventually, at time $t = n$, the engagement will terminate and either restart, as in robot soccer after a goal is scored, or end permanently, as in after one side has surrendered or retreated after a military engagement. In either case, it is natural to ask questions about the dynamics of the engagement. Why does the game state trajectory behave the way it does? Is there an underlying, application-agnostic theory that can explain what is observed empirically? Can any such theory provide insights for developing better forward models of such engagements? As an initial step towards answering these questions, we introduce two concepts: the *strategy manifold*, and the *strategy gradient*.

4.1 The Strategy Manifold

Consider a robot soccer game played by two relatively equal, proficient teams. Here, the observable state factors are the position and velocity of each robot and the position and velocity of the ball. Much like matches played by professional humans, the number of states technically allowed by the rules is much greater than the number of states visited by the two teams during competition. For example, it is perfectly legal for all players to move into a single quadrant of the field, but this configuration is never seen in real games. Not rarely, *never*. This distinction is important since in many cases it allows us to simplify the models or representations of the game without sacrificing any model accuracy.

In our robot soccer example, observable state factors exist in $\mathbb{R}^N$, where $N = (2n) \times 6 + 4$ and n is the number of robots on each of the two teams. The factor of 6 comes from each robot having position (x, y, θ) and velocity $(\dot{x}, \dot{y}, \dot{\theta})$ state factors. The added 4 is due to the ball's state, which has no orientation (θ) term. Randomly walking through this space, one would expect very few repeated (to within some ϵ) states due to the high dimensionality. However, we see a small fraction of states visited over and over again during real games. We hypothesize that the limited exploration of the observable state space is the manifestation of a structure imposed on the observable state factors by the hidden state factors.

Intuitively, this hidden structure can be described by a manifold $\mathcal{M} \subseteq \mathbb{R}^N$. For example, consider a game where the state space is defined in $\mathbb{R}^2$, and the rules of the game technically allow any (x, y) state as a valid game state. However, if one or more rules of the game have an indirect effect such that winning strategies only need to visit states for which the relationship $x^2 + y^2 = 1$ is true, then the effective state space for which an agent needs to model its opponent can be reduced from all of $\mathbb{R}^2$ to some manifold $\mathcal{M} \subset \mathbb{R}^2$, which in this case describes the unit circle. Clearly, understanding this manifold has many advantages, dimensionality reduction and informed choice of basis (feature set) chief among them.

Definition 4. An adversarial game with N observable state factors represented by a game state Y^N has a **strategy manifold** $\mathcal{M}(\mathcal{R}, \mathcal{C}, s_0, \ldots, s_n) \subseteq Y^N$ where:

$\mathcal{R}$: The rules of the game.

$\mathcal{C}$: The capabilities of the agents.

s_i : The strategy or policy of agent i.

Strategy manifolds describe all the reachable states of a particular game, given the rules, capabilities of the agents, and the respective strategies of each team. Again, consider a robot soccer match. The rules dictate the size of the field, prescribe a maximum kick speed, and prohibit certain types of robot-robot collisions among other things. These rules constrain all possible game state observations in $\mathbb{R}^N$ to some subset $\mathcal{M}_\mathcal{R} \subset \mathbb{R}^N$ which are allowed by the rules. Next, we apply the inherent limitations of the agents; the soccer robots have some maximum achievable acceleration and velocity. Note that the rules do not explicitly limit these attributes. Rather the limitations are due to the limited capabilities of the agents themselves. Note also that there may be indirect relationships between the rules and the capability displayed by agents. For instance, a robot with low acceleration ability and high maximum velocity may never achieve its maximum velocity on a small field since the robot needs long distances to accelerate all the way up to maximum velocity. Thus, after considering both the rules $\mathcal{R}$ and the agents' capabilities $\mathcal{C}$, we see the possible game state observations are further constrained to $\mathcal{M}_{\mathcal{R},\mathcal{C}} \subset \mathcal{M}_\mathcal{R} \subset \mathbb{R}^N$.

Since strategies may be arbitrarily bad, $\mathcal{M}_{\mathcal{R},\mathcal{C}}$ is the tightest constraint possible on the state space for any particular game, given arbitrary competitors. For example, consider professional soccer leagues and the strategies they employ (long passes and players spread out over the whole pitch) compared to children's leagues, where the dominant strategy is for all players to run towards the ball and attempt to kick it in the direction of their opponent's goal. However, if we know at least one of the team's strategies, we can make even stronger statements about the characteristics of the strategy manifold by constraining the values of variables controllable by the team whose strategy we know. Since in opponent modeling information about one's own team is often available, this is often exploitable. If we let s_0 be the strategy of the control team, we get $\mathcal{M}_{\mathcal{R},\mathcal{C},s_0} \subset \mathcal{M}_{\mathcal{R},\mathcal{C}} \subset \mathcal{M}_\mathcal{R} \subset \mathbb{R}^N$.

Although most opponent modeling research focuses on predicting opponent behavior or approximating opponent strategy, the concept of the strategy manifold can in principle be used to learn any singular held out piece of information. For instance, given the strategies and capabilities of all agents, the rules of the game should be learnable through the same manifold construction process. Similarly, given strategies and rules, agent capabilities

should learnable. However, we hypothesize that formulations for learning rules and agent capabilities may be more challenging than simply learning an opponent's strategy, since it is generally not possible to assume that given strategies are optimal, whereas it is usually safe to assume priors on rules or agent capabilities are accurate.

The utility of the strategy manifold is clear for policy approximation approaches, since it allows better priors for the transition function, but it may also aid discriminative and generative opponent models. The quality of Z as an abstraction space and $g()$ as an abstraction mechanism is related to the ability of states $z \in Z$ to both map to strategically distinct regions of the strategy manifold $\mathcal{M} \subset Y$ as well as minimize the number of game states $y \in Y$ which are not in the codomain of $g^{-1}()$. Understanding the extra restrictions on possible inputs to $g()$, provided by knowing the strategy manifold $\mathcal{M}$, can help us design better methods for $g()$ or even help us determine Z. Making good use of offline compute resources to validate abstractions for these two properties is something we think will dramatically increase the performance of many opponent modeling systems, and the problem of determining the strategic uniqueness of $z \in Z$ in particular seems like an opportunity for unsupervised algorithms to excel. However, even if Z is chosen well, that is, the compressed representation of the opponent's state generally has these properties, effective opponent modeling still requires accurate prediction, denoted earlier by the function $h()$. For this, we introduce the concept of the strategy gradient.

4.2 The Strategy Gradient

While the strategy manifold restricts the set of reachable states given the rules, agent capabilities, and strategies of the agents, it does not explain how the game state might evolve from a given starting point. For this, we need the concept of the strategy gradient.

Let $V_i(y; s_i, s_j)$ be a function which maps any game state to a real value in the interval $[-\kappa, \kappa]$, and suppose $V_i = -\kappa$ for any $y \in Y$ in which team i wins and $V_i = \kappa$ for any $y \in Y$ in which team i loses. Values between $-\kappa$ and κ may also represent terminal states in adversarial situations with more than two outcomes. In a deterministic game without an adversary, or with an adversary that never takes any actions, such as an opponent soccer team whose players never move, prediction of future game states could be achieved for arbitrary future times by following the negative gradient of V_i, $-\nabla V_i$, until a local minimum is reached. In a stochastic version of the same game, predicting a distribution over future game states could be achieved by running repeated trials of gradient descent, moving along $-\nabla V_i$ and adding an additional noise term $\eta(\sigma)$ at each step, where σ parameterizes the stochasticity. In this case, the trajectory evolves by $-\nabla V_i^{\sigma} = -\nabla V_i + \eta(\sigma)$ every action.

Now, let $V^* = V_i - V_j$, where agents or teams i and j are competing against each other. We call V^* the complete game value function.

Definition 5. Given a complete game value function V^*, the **strategy gradient** is ∇V^*. The **partial strategy gradient** is $\nabla_i V^* = \left[\frac{\partial V^*}{\partial y_1^i}, \ldots, \frac{\partial V^*}{\partial y_n^i}\right]^T$, where $y^i \in Y$ are game state variables which team i can control.

In this case team i would attempt to perform gradient descent on V^*, while team j would attempt to perform gradient descent on $-V^*$. However, usually agents cannot directly affect all meaningful state variables, such as the position of the opponent goaltender

in robot soccer. So, they will follow their respective partial strategy gradients. Moreover, in many domains $\nabla_i V^* \cdot \nabla_j V^*$ is often 0, except when both teams have the opportunity to simultaneously affect some neutral part of the environment or directly affect their adversaries. For instance, in robot soccer, when one team has the ball, the other team typically cannot directly affect the location or velocity of the ball. The only time both teams can affect the ball's state is when both teams have a player who is in range to kick it.

If the payoff and dynamics depend on a set of continuous, observable, controllable variables, and the dynamics and reward are isotropic, then conceptually, the saddle points of the strategy gradient are a continuous analog of local Nash equilibria for the given encounter. That is, assuming a starting game state within some basin of attraction, the opponents would mutually seek the same global game state. The general trajectory of the entire game state can thus be described by following the strategy gradient, and there are likely small modifications one could make to extend this analogy to conditions beyond purely continuous variables. Of course due to stochastic actions, imperfect information, or changing strategies, it is not guaranteed that a particular saddle point would be reached even with an initial state within the basin of attraction.

This formulation presents two challenges. First, how can we compute V^* and ∇V^* without full knowledge of the opponent's strategy and given limited observations? Second, if the game is stochastic, how can we learn or model σ and $\eta(\sigma)$? To compute V^* and its gradient exactly seems intractable, or at the very least impractical. However, inverse reinforcement learning and other reward or value function approximation techniques seem to be reasonable first steps. Additionally, estimates for V^* are useful as long as the gradient points roughly in the right direction, even if the magnitude is wrong.

Estimating σ and $\eta(\sigma)$ will require substantial domain-specific knowledge and engineering. Many agents already use models of their own uncertainty with respect to different actions, which will be readily available to bootstrap these calculations. Estimating these quantities for opponents will require new ground to be broken in the subfield of skill estimation, and depending on the domain, may also require new methods for automatic action identification or labeling. Figure 3 illustrates an example strategy manifold and gradient.

As with the strategy manifold and the definitions of abstraction types, strategy gradients have been explained and defined assuming continuous variables. However, most real-world systems will have a mixture of discrete and continuous variables. There are several ways to handle this. The most straightforward would be to simply assign scalar values to these variables, and although the intermediate values may not be realized, the underlying structure and gradient will still be represented. One can always use the finite difference method where needed instead of a continuous derivative. A more unsupervised approach might be to create vector-space embedding representations of discrete variables, as is often done using a neural network.

5. Opponent Modeling Evaluation Framework

The classifications discussed in section 3 are useful semantically, and provide some intuition for how to design opponent modeling systems under different constraints. However, the question of how to compare approaches is still an open one. Although opponent modeling in games featuring some or all of the properties outlined in the introduction has been a

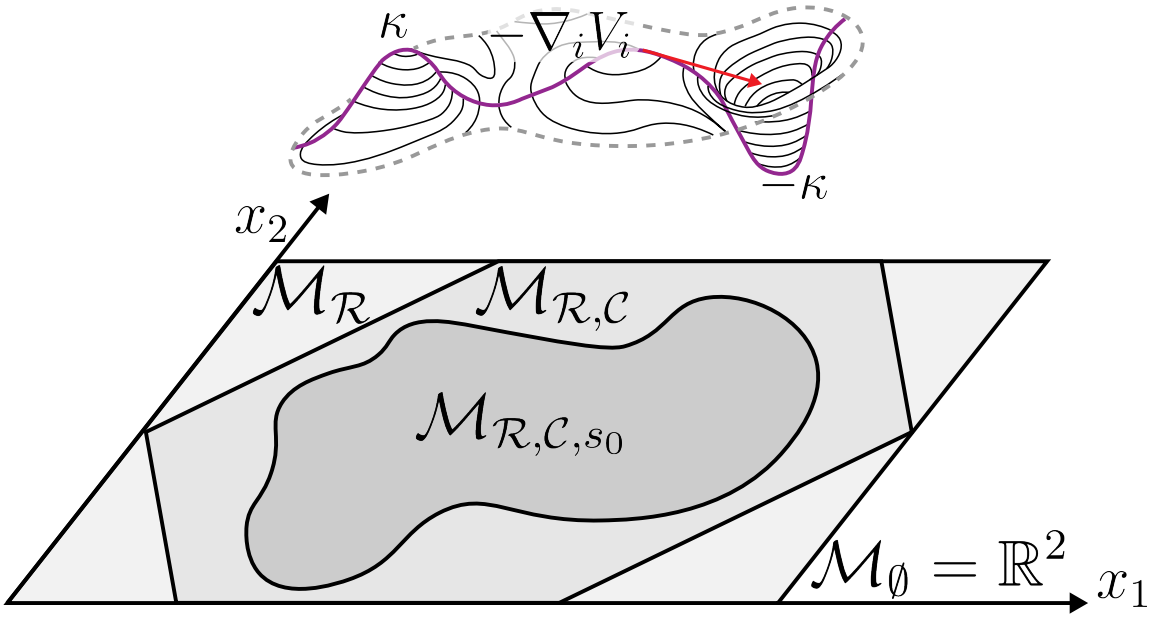

Figure 3: An illustration of a hypothetical game state and several increasingly restrictive versions of a strategy manifold. In the top part of the figure, level curves are shown representing the value for agent i of every part of the strategy manifold $\mathcal{M}_{\mathcal{R},\mathcal{C},s_0}$. The purple line travels along the value function for the manifold at some constant value for x_2. If agent i can only affect $x_1 \in Y$, then the tangent line to the purple curve represents the strategy gradient, shown here at at arbitrary point along the line in red.

serious research topic for decades, there has been a persistent lack of consensus regarding the definition of algorithm deliverables and the formulation of independence assumptions. Here, we address this by proposing a framework for discussing the deliverables and independence assumptions in opponent modeling approaches.

Every abstraction level lends itself to a particular suite of algorithmic techniques, but all approaches are after the same fundamental information: the probability of an opponent being in some particular state at some time in the future, given some domain knowledge, some past experiences, and the current state. Formally, for a game beginning at t_0 and currently at time t, where a team of n partially observable agents opposes a team of m fully observable agents, we write this as

$$P(O^{1:n}_{t_p} | O^{1:n}_{t_0:t}, C^{1:m}_{t_0:t}, C^{1:m}_{(t+1):t_p}, E_{t_0:t}, E_{(t+1):t_p}, K, T), \tag{2}$$

where t_p is the desired time of prediction. $O^{1:n}$ and $C^{1:m}$ are the joint states of the opponent and controlled team, respectively. E is the state of the environment, such as field boundaries, offsides lines, score, time on the clock, *etc.* K is any expert domain knowledge, and T is any transferred knowledge from other similar games. We explicitly distinguish between past and future controlled team states and environment states because of the common, somewhat dubious assumptions that opponents will not change course of action even in the presence of changing behavior by the control team, and that certain future aspects of the environment

are highly unpredictable. Since many methods do not consider the terms $C^{1:m}_{(t+1):t_p}$ and $E_{(t+1):t_p}$, but do consider $C^{1:m}_{t_0:t}$ and $E_{t_0:t}$, we feel this is a more convenient formulation.

We propose this notation because it is general, in that all opponent modeling techniques can be represented by such an expression, and explicit, because it references the underlying data being used to train the algorithm, rather than intermediate representations which vary from technique to technique. This notation also explicitly includes several variables related to system performance which are often obfuscated by algorithmic details. For example, we would prefer algorithms which can predict accurately farther into the future to those which can only predict a short time into the future. This metric is explicit in the given expression, and is represented by the quantity $t_p - t$. Another easily accessible axis of comparison is the size of the set of variables being predicted relative to the size of all predictable variables. If the prediction regards variables $O^{1:x}_{t_p}$, it is clear from the expression whether $1:x$ represents a single agent, multiple agents, or the entire team. Moreover, stating deliverables in terms of this expression may also facilitate more careful experimentation and exposition related to the functions $g^{-1}()$ and $h()$ from section 3.3, since many papers do not state how successful classification of roles or tactics translates back into belief about the likelihood of future game states. Lastly, independence assumptions are easy to compare, as determining the information which a given system operates on is trivial in this framework.

We now compare a number of studies and their proposed techniques across several dimensions, including the types of features used, the learning algorithms employed, the level of abstraction at which the predictive component operates, and the equivalent expression for the work's formulation as given by the framework we just introduced. Tables 2, 3, 4, and 5 present comparisons of discriminative, generative, policy approximation, and other approaches, respectively. Table 1 is a reference for many of the acronyms in the tables. Note that variables O^n represent predictions for future states of an entire group or team, while O^k represents predictions for a subset of the group or team where $k \leq n$.

Acronym	Definition	Acronym	Definition
AHMM	Abstract Hidden Markov Model	NN	Neural Network
CBR	Case-based Reasoning	PCA	Principal Component Analysis
DBBR	Deviation-Based Best Response	PFA	Probabilistic Finite Automata
DFA	Deterministic Finite Automata	PI-POMDP	Parameterized Interactive POMDP
DS(D)SF	Domain-specific (Dis)Similarity Function	RANSAC	RANdom Sample And Consensus
DTIM	Discriminative Temporal Interaction Manifold	RBPF	Rao-Blackwellized Particle Filter
EDAhR	Expert Designed Ad-hoc Rules	RL	Reinforcement Learning
EDPM	Expert Designed Payoff Matrix	RSF	Raw State Factors
EM	Expectation Maximization	SMDM	Semi-Markov Decision Process
GMM	Gaussian Mixture Model	SoA	Sequence of actions
HAC	Hierarchical Agglomerative Clustering	SoMF	Sequence of Metric Features
HMM	Hidden Markov Model	SoP	Sequence of Positions
HMMP	Hierarchical Multiagent Markov Processes	SoSAP	Sequence of state-action pairs
MA	Manual Annotation	SSG	Stackelberg Security Games
MDP	Markov Decision Process	SUQR	Subjective Utility Quantal Response
MFS	(ED)Model Forward Simulation	SVM	Support Vector Machine
MKL	Multi Kernel Learning	TM	Template Matching

Table 1: A key from acronyms to jargon, required for compact tables to follow.

Work	Type of Feature	Learning Framework	Abstraction: Def O^n	Prediction Expression
(Ahmadi et al., 2003)	SoMF	Hierarchical CBR	Agent State $x, \dot{x}$ (A, B, C)	$P(O^n_{t+\Delta t}\|O^n_{0:t})$
(Ledezma et al., 2009)	SoA	Decision / Regression Tree	Agent Action (A)	$P(O^1_{t+1}\|O^n_{0:t})$
(Spronck & den Teuling, 2010)	SoA	Support Vector Machine	Player Strategy (C)	$P(O^n_{t+1}\|O^n_{0:t})$
(Weber & Mateas, 2009)	SoA	Case-based Reasoning	Player Strategy (C)	$P(O^n_{t+1}\|O^n_{0:t})$
(Bombini et al., 2010)	SoA via EDAhR	kNN + Tanimoto Measure	Team Identity (C)	$P(O^n_{t+1}\|O^n_{0:t}, K)$
(Kaminka et al., 2002)	SoA via EDAhR	Dependency Detection	Team Identity (C)	$P(O^n_{t+1}\|O^n_{0:t}, K)$
(Hsieh & Sun, 2008)	SoA	Case-based Reasoning	Agent Strategy (C)	$P(O^n_{t+\Delta t}\|O^n_{0:t})$
(Farouk et al., 2017)	SoA via EDAhR	Case-based Reasoning	Agent Strategy (C)	$P(O^n_{t+\Delta t}\|O^n_{0:t}, K)$
(Laviers et al., 2009)	SoP	Support Vector Machine	Football Play (C)	$P(O^n_{t+1}\|O^n_{t_0:t})$
(Laviers & Sukthankar, 2011)	SoP	Support Vector Machine	Football Play (C)	$P(O^n_{t+1}\|O^n_{t_0:t})$
(Siddiquie et al., 2009)	SoP, play type via MA	SVM + MKL	Football Play (C)	$P(O^n_{t+1}\|O^n_{t_0:t})$
(Sukthankar & Sycara, 2007)	SoA	SVM + Dempster-Shafer	Agent Role (A)	$P(O^1_{t+1}\|O^1_{t_0:t})$
(Beetz et al., 2006)	SoP, SoMF, MA	Decision Tree	Agent Action (A)	$P(O^1_{t+1}\|O^1_{t_0:t})$
(Riley & Veloso, 2000)	SoP, SoA	Decision Tree	Team Strategy (C)	$P(O^n_{t+1}\|O^n_{t_0:t})$
(Nair et al., 2004)	SoP, SoA	Decision Tree + PFA	Strategy / Role (A,B,C)	$P(O^n_{t+1}\|O^n_{t_0:t})$
(Kamrani et al., 2016)	SoP, SoA, SoMF	Decision Tree + NN	Military Tactic (A,B,C)	$P(O^k_{t+1}\|O^n_{t_0:t})$
(Fukushima et al., 2017)	Current Positions	NN, SVM, Random Forest	Team Strategy (C)	$P(O^n_{t+1}\|O^n_t)$
(Sadilek & Kautz, 2010)	RSF	Markov Logic Network	Subgroup Activity (B,C)	$P(O^k_{t+1}\|O^n_{t_0:t})$
(Vail et al., 2007)	RSF, SoMF via EDAhR	Conditional Random Fields	Agent Role (A)	$P(O^1_{t+1}\|O^n_{t_0:t}, K)$
(Steffens, 2002)	SoA, SoP	Case-based Reasoning	Team Strategy (C)	$P(O^n_{t+1}\|O^n_{t_0:t})$
(Steffens, 2005)	SoP, SoMF via EDAhR	CBR + DSSF	Team Strategy (C)	$P(O^n_{t+1}\|O^n_{t_0:t}, K)$
(Wendler & Bach, 2003)	SoMF via EDAhR	CBR + DSSF	Subgroup Activity (B,C)	$P(O^k_{t+1}\|O^n_{t_0:t}, K)$
(Lucey et al., 2013)	SoP, SoA via MA	Linear Discriminant Analysis	Team Identification (C)	$P(O^n_{t+1}\|O^n_{t_0:t})$
(Schadd et al., 2007)	SoA	Hierarchical Fuzzy Models	Atk/Def, Unit Type (C)	$P(O^n_{t+1}\|O^n_{t_0:t})$
(Biswas et al., 2014)	RSF	TM using DSSF	Current Agent Tactic (A)	$P(O^1_{t+1}\|O^n_t, C^n_t, C^n_{t+1}, K)$
(Devaney & Ram, 1998)	Pair-wise SoP	TM using DSSF	Subgroup Activity (B)	$P(O^k_{t+1}\|O^n_{t_0:t}, K)$
(Perše et al., 2009)	SoA via GMM, EDAhR	Levenstein Distance	Offensive Team Play (C)	$P(O^n_{t+1}\|O^n_{t_0:t}, K)$
(Li et al., 2009)	SoP, play type via MA	DTIM + EM	Football Play (C)	$P(O^n_{t+1}\|O^n_{t_0:t})$

Table 2: Discriminative approaches

Work	Type of Feature	Learning Framework	Abstraction: Def O^n	Prediction Expression
(Riley & Veloso, 2001)	SoP	Naive Bayesian Network	Agent Position (A)	$P(O^1_{t+1}\|O^n_{0:t})$
(Stankiewicz & Schadd, 2009)	SoA	Bayesian Inference	Agent Rank (A)	$P(O^1_{t+1}\|O^n_{0:t})$
(Synnaeve & Bessiere, 2011)	SoA via EDAhR	Bayesian Networks	Team Strategy (C)	$P(O^n_{t+1}\|O^n_{t_0:t}, K)$
(Kuhlmann et al., 2006)	SoMF	Linear Regression	Team Formation (B, C)	$P(O^n_{t+1}\|O^n_{t_0:t})$
(Shen & How, 2019a)	SoA	MDP + NN	Agent Role (A)	$P(O^1_{t+1}\|O^1_{t_0:t})$
(Torkaman & Safabakhsh, 2019)	SoA	Bayesian Networks	Team Strategy (C)	$P(O^n_{t+1}\|O^n_{0:t})$
(Fagan & Cunningham, 2003)	SoA	CBR + Plan Library	Agent Plan (A)	$P(O^1_{t+\Delta t}\|O^1_{0:t})$
(Ball & Wyeth, 2003)	SoP	Naive Bayesian Network	Agent Role (A)	$P(O^1_{t+1}\|O^n_t)$
(Leece & Jhala, 2014)	SoP, SOA	Markov Random Field	Army Size (C)	$P(O^n_{t+1}\|O^n_{0:t})$
(Wei et al., 2013)	SoP, SoA	Bayesian Network + GMM	Success of Action (A)	$P(O^1_{t+1}\|O^n_{t_0:t})$
(Intille & Bobick, 1999)	SoP, SoMF via EDAhR	Bayesian Network	Football Play (C)	$P(O^n_{t+1}\|O^n_{t_0:t}, K)$
(Intille & Bobick, 2001)	SoP, SoMF via EDAhR	Bayesian Network	Football Play (C)	$P(O^n_{t+1}\|O^n_{t_0:t}, K)$
(Han et al., 2000)	SoP	Hidden Markov Models	Agent Action (A)	$P(O^1_{t+1}\|O^1_{t_0:t})$
(Bui et al., 2002)	SoP or SoA	AHMM + RBPF	Policy Hierarchy (A)	$P(O^1_{t+\Delta t}\|O^1_{t_0:t})$
(Saria & Mahadevan, 2004)	SoP or SoA	HMMP + RBPF	Policy Hierarchy (A,B,C)	$P(O^n_{t+\Delta t}\|O^n_{t_0:t})$
(Sukthankar & Sycara, 2006)	SoP, SoMF via EDAhR	HMM + RANSAC	Subgroup Activity (B)	$P(O^k_{t+1}\|O^n_{t_0:t}, E_{t_0:t}, K)$
(Floyd et al., 2017)	SoMF via EDAhR	CBR + DSSF	Team/Agent Role (A,B,C)	$P(O^k_{t+1}\|O^n_{t_0:t}, K)$
(Butler & Demiris, 2009)	RSF, EDMFS	Simulation + DSSF	Atk/Def, Formation (B,C)	$P(O^n_{t+1}\|O^n_{t_0:t}, K)$

Table 3: Generative approaches

In all tables each row features one publication, and the features used, learning or modeling framework, abstraction level of the prediction target, and the equivalent prediction expression are presented in the following columns from left to right. From the tables we can see that there does not appear to be any particular algorithm that has enjoyed prolonged success in any domain, with most learning frameworks showing up and disappearing as they become more or less popular in machine learning in general. This suggests that meaningful progress towards better opponent modeling systems may require innovation on multiple fronts and will not be possible via improvement in machine learning techniques alone. Also, somewhat surprisingly, we see strong preference in discriminative approaches towards using

Work	Type of Feature	Learning Framework	Abstraction: Def O^n	Prediction Expression
(Lockett et al., 2007)	SoA	SharpNEAT + NN	Agent Action (A)	$P(O^1_{t+1}\|O^n_t)$
(Synnaeve et al., 2018)	SoP	Recurrent NN	Agent Position / Type (A)	$P(O^1_{t+\Delta t}\|O^n_t, E_{0:t})$
(Baez, 2015)	SoP	Markov Chain	Next Position (A)	$P(O^1_{t+1}\|O^n_t)$
(Visser & Weland, 2002)	SoMF via EDAhR	Decision Tree	Agent Action (A)	$P(O^1_{t+1}\|O^1_{t_0:t}, K)$
(Markovitch & Reger, 2005)	SoSAP	Decision Tree	Opp. Weakness (A,B,C)	$P(O^n_{t+1}\|O^n_{t_0:t})$
(Kar et al., 2017)	SoA or SoP	Decision Tree Ensemble	Poaching Targets (A,B,C)	$P(O^n_{t+1}\|O^n_{t_0:t})$
(He et al., 2016)	SoSAP	Deep RL	Agent Action (A)	$P(O^1_{t+1}\|O^1_t)$
(Hong et al., 2017)	SoSAP, RSF	Deep RL	Agent Action (A)	$P(O^1_{t+1}\|O^1_{t-M:t})$
(Hoang et al., 2017)	RSF, EDAhR	Mac-Dec-POMDP	Team Strategy (C)	$P(O^n_{t+1}\|O^n_{t_0:t}, K)$
(Wunder et al., 2011)	SoA	PI-POMDP	Strategy Level (C)	$P(O^n_{t+1}\|O^n_{t_0:t})$
(Yin et al., 2016)	SoP, RSF	SMDM + RBPF	Agent Destination (A)	$P(O^1_{t+\Delta t}\|O^1_{t-s:t})$
(Fard et al., 2007)	SoA	DFA learning + EDPM	Next Agent Action (A)	$P(O^1_{t+1}\|O^n_{t_0:t}, K)$
(Rovatsos et al., 2003)	SoA	DFAs via US-L^* + DSSF	Strategy Class (A,B,C)	$P(O^n_{t+1}\|O^n_{t_0:t}, K)$
(Marín et al., 2005)	SoP, SoMF via EDAhR	DFAs via US-L^* + DSSF	Agent Position (A)	$P(O^1_{t+\Delta t}\|O^n_{t_0:t}, E_t, K)$
(Chakraborty et al., 2013)	SoA	DSSF	Agent Actions (A)	$P(O^1_{t+\Delta t}\|O^1_{t-L:t}, K)$
(Ganzfried & Sandholm, 2011)	SoA	DBBR	Strategy (C)	$P(O^n_{t+1}\|O^n_{t_0:t})$
(Yang et al., 2014)	SOA or SOP	SSG + MLE using SUQR	Poaching Targets (A,B,C)	$P(O^n_{t+1}\|O^n_{t_0:t})$
(Stone et al., 2000)	RSF, Opp. Action Set	Sim. of Opp. Actions	Success of Action (A)	$P(O^1_{t+\Delta t}\|O^1_t)$
(Iglesias et al., 2009)	SoA via EDAhR	χ^2-test on Trie Traces	Anomolous SOA (A,B,C)	$P(O^n_{t+1}\|O^n_{t_0:t}, K)$

Table 4: Policy Approximation approaches

Work	Type of Feature	Learning Framework	Abstraction: Def O^n	Prediction Expression
(Hayes & Beling, 2018)	SoA	HAC	Team Strategy (C)	$P(O^n_{t+1}\|O^n_{t_0:t})$
(Lattner et al., 2005)	SoA, SoMF	HAC + DSDF	Agent Action (A)	$P(O^1_{t+1}\|O^n_{t_0:t}, K)$
(Cliff et al., 2013)	SoP	Shannon Info. Dynamics	Activity Correlation (B)	$P(O^k_{t+1}\|O^n_{t_0:t}, C^n_{t_0:t})$
(Cliff et al., 2017)	SoP	Shannon Info. Dynamics	Activity Correlation (B)	$P(O^k_{t+1}\|O^n_{t_0:t}, C^n_{t_0:t})$
(Erdogan & Veloso, 2011)	SoP via EDAhR	HAC + Hausdorff metric	Pair Trajectories (B)	$P(O^n_{t+\Delta t}\|O^n_{t_0:t}, K)$
(Yasui et al., 2013)	SoP	HAC + DSDF	Team Strategy (C)	$P(O^n_{t+1}\|O^n_{t_0:t}, K)$
(Adachi et al., 2016)	SoA via EDAhR	HAC + DSDF	Team Strategy (C)	$P(O^n_{t+1}\|O^n_{t_0:t}, K)$
(Trevizan & Veloso, 2010)	SoMF via EDAhR + PCA	Frobenius Norm	Defensive Strategy (C)	$P(O^n_{t+1}\|O^n_{t_0:t}, C^n_{t_0:t}, K)$

Table 5: Other approaches

complete abstractions, whereas that trend is opposite for policy approximation approaches, where atomic abstractions are preferred. Generative approaches seem to be split roughly equally. This may be due to the relatively large state space required to describe an entire team, which typically poses many more challenges for policy approximation schemes than for discriminative or generative ones.

We also see two trends related to potential sources of information which are rarely taken advantage of and which point perhaps to new opportunities for opponent modeling research. First, very few techniques are explicitly modeling the environment variables. Environment variables in robot soccer may be variables such as the overall score, the amount of time left in the game, or whether a robot has received a caution. Environment variables in other settings might include terrain or visibility constraints, or the status of an objective which can only be captured or lost over several encounters. These variables clearly affect the strategy of human soccer agents who, for instance, may intentionally waste time or play very defensively near the end of a game if they are winning, eliminating chances for their own team to score, but also drastically reducing the probability of their opponent scoring. Currently, very few approaches, even under perfect conditions, could predict this behavior.

Similarly, there is a term which is not just rare, but altogether absent from the literature. We did not encounter any approach which attempted to use transfer learning to bootstrap opponent modeling across different domains. We hypothesize that transfer learning will not only be an important component for achieving more performant opponent modeling

systems, but it may also offer an invaluable tool for examining what exactly systems are learning. We expect similar games to support similar classes of effective strategies, and we expect opponent modeling systems to find these strategies. The robustness of all of these assumptions can be tested in part by experimenting with transfer learning.

We should also add that virtually all approaches use expert domain knowledge in some form or another. Here, we add the K term explicitly into the probability expressions if the technique uses distance functions or feature extraction rules which are engineered specifically for that domain and would not have meaning in other domains.

6. Open Problems and Future Research Directions

Opponent modeling researchers have made significant progress in the last two decades. However, there are still many opportunities for improvement, including a greater focus on automatic feature selection and validation, more comparisons between methods facilitated by frameworks which unify vocabulary and establish metrics, and development of methods that generalize to different games which share strategic concepts. Given these high-level aspirations, we identify six key areas and approaches which seem particularly ripe for investigation in the context of opponent modeling in RoboCup and other similar domains.

6.1 Modeling Failed Actions

Correctly identifying actions is critical to many opponent modeling systems. However, few methods are capable of modeling failed actions. Failed actions provide arguably as much information as successful actions, yet to the best of our knowledge there do not exist any opponent modeling approaches which model failed actions either via explicitly reasoning about failure, or by calculating distributions over the action which was attempted. Both identifying what an opponent intended, as well as how to incorporate that information into a learning algorithm, are open research questions. Identifying failed actions and modeling opponent capabilities are closely related. Some work exists which models opponent search depth in a minimax setting (Carmel & Markovitch, 1993, 1996a; Donkers et al., 2001), but analogs for adversarial domains with partial observability, and continuous or stochastic actions do not yet exist.

Moreover, data about failed actions and models of opponent capabilities can be mined to improve estimates of the strategy gradient and the reliability and stochasticity of certain actions for the given opponent. Much like many basketball and soccer teams are content to let the opposing team shoot from long range since the probability of scoring is low, we would like our opponent modeling systems to reason similarly not only about what the opponent plans to do, but also about their probability of success should they choose a given action. The problem of skill estimation (Archibald & Nieves-Rivera, 2018) is closely related, and Bayesian techniques have been proposed for simulated, real-valued games including darts and billiards (Archibald & Nieves-Rivera, 2019), but this is still an under-explored area of research and has yet to be integrated in opponent modeling systems at any notable scale.

6.2 Using Model Inference Techniques from Software Engineering

Although many, many learning algorithms have been applied to opponent modeling, there is one class of approaches which are notably absent. In software engineering, there is a subfield of study dedicated to building models of programs given only the stack trace of the program or the value of variables at given points in program execution. Thus, the model does not have access to the source, only a (perfect) black-box simulation. This problem statement has a natural mapping to opponent modeling.

Two dominant techniques have emerged to address this problem. One, proposed originally in 1972, uses stack traces organized into trees. The trees are compressed via the k-tails algorithm (Biermann & Feldman, 1972), resulting in a state machine which approximates the program in question. Modern variants use more complicated merging and compression algorithms (Heule & Verwer, 2013) and form hierarchical models (Alimadadi et al., 2018). The second technique mines program invariants, such as an argument to a function always being positive, from the execution traces in order to build a specification of the program. The original version, Daikon (Ernst et al., 2007), has a number of variants. See (Krka et al., 2014) for an overview of invariant-based model inference techniques.

6.3 Relaxing the Static Strategy Assumption

One popular assumption in opponent modeling is that the opposing team follows a fixed strategy. This assumption is often required in order to make the problem tractable, but can severely limit the applicability and accuracy of the system. Strategies violating this assumption are known as non-stationary strategies and may involve scheduled changes between multiple fixed strategies or continually updating strategies as the agent experiences and learns from its environment. We hypothesize that some limited form of reasoning about the opponent's ability to change strategy or to react to actions taken by the control team may lead to greater prediction accuracy and greater applicability to systems where modeling a dynamic opponent strategy is necessary, and this has very recently been supported experimentally (Yu et al., 2021).

Of the directions suggested here, this is the most mature. Already, there has been substantial work on dealing with non-stationarity in both game theory (Hernandez-Leal et al., 2017a, 2017b), where the problem is often cast as identifying an agent's choice between fixed strategies, and reinforcement learning (Papoudakis et al., 2019; Li et al., 2021), where most agents are assumed to be continually updating their strategies. However, these approaches and their possible interactions in a multi-agent setting give rise to further questions. In particular, new research is needed to understand when and how the theory of mind (Premack & Woodruff, 1978; Goldman et al., 2012) can and should be applied. Some initial work has shown that for some games, first and second order theory of mind has substantial benefit, with higher orders having decreasing return on investment (De Weerd et al., 2013; Tian et al., 2021b). Recently, new graphical models have been proposed, including the threatened MDP (Gallego et al., 2019) and a Bayes theory of mind on policy (Yang et al., 2019), which allow reasoning about an adversarial agent's planning process, and whether it may itself be reasoning about a theory of mind.

Clearly, this problem quickly becomes intractable. One possible way to mitigate this problem in general is to combine these models with Monte Carlo methods, which has been

tried in related domains (Yin et al., 2016; Bard & Bowling, 2007). Monte Carlo methods have the advantage of being able to leverage a wealth of successful research in sampling and simulation (Browne et al., 2012; Liu et al., 2018), and we hypothesize that the biggest challenges will be to parameterize strategies both in terms of their policy as well as the reliability of stochastic actions such that sampling approaches can adjust parameters online in order to fully explore multiple hypothetical opponent responses.

6.4 Unsupervised and Semi-Supervised Learning for Feature Selection

Feature selection is often somewhat of a dark art in machine learning. However, we hypothesize that many opponent modeling problems contain structure that can be exploited during the feature selection process. In particular we believe semi-supervised and unsupervised learning techniques, including manifold learning, are particularly promising because they allow some generality across applications and because they greatly reduce the effort required to replicate results. Feature extraction, broadly speaking, is well-studied (Ding et al., 2012; Storcheus et al., 2015), and continues to be an active area of research. Various classes of techniques have been proposed, including semi-supervised learning (Zhu, 2005; Sheikhpour et al., 2017), where only a subset of data instances are labeled. In particular, in light of the argument made for the utility of the strategy manifold and similar concepts, research on manifold learning (Izenman, 2012; Huo et al., 2007) offers a promising path forward, and indeed has already become a mainstay in other fields (Pless & Souvenir, 2009).

The raw data from RoboCup and similar games resides in an enormous state space. However, when two teams play against each other, the states actually visited represent a small fraction of this space. According to the idea of the strategy manifold this is due to the interaction between agent capabilities and the rules of the contest, which give the state space an underlying structure and cause teams to converge toward some subset of states. It is not a coincidence, for instance, that professional soccer teams most often play with 3-4 defenders, 4-5 midfielders, and 1-2 attackers, even though all possible assignments of roles to the 10 outfield players are allowed by the rules. We hypothesize that understanding this structure is a prerequisite to optimal feature selection, and we advocate for greater application of both traditional unsupervised dimensionality reduction techniques, as well as semi-supervised approaches.

In fact, evidence of this underlying structure already exists within opponent modeling research, albeit indirectly. Research by Gold (2010), on developing a goal detection system for a video game using an input-output hidden Markov model (IOHMM) (Bengio & Frasconi, 1995), uncovered an interesting phenomenon. By directing players to achieve particular goals and recording their sequence of actions, both a general model as well as a personalized model could be constructed. These personalized models effectively predicted behavior of experienced players. However, the personalized models actually decreased overall performance for novice players. We hypothesize that this is because novice players exhibit something closer to a random walk through the game state space, and thus the underlying strategy manifold they access is not described well by the general model. Furthermore, the discovery of strategy 'levels' by Wunder et al. (2011) in the Lemonade Stand Game supports the idea that some regions of game-state space may only be visited during interactions between particular, perhaps unique, strategies.

6.5 Estimating Trajectories via Strategy Gradients

Techniques to use domain knowledge in a principled way to define abstractions are still an active area of research. In terms of system effectiveness, perhaps the most important decision in modeling an opponent is: "At which level of granularity should predictions be made?" This is equivalent to "What is the definition of $O_t^{1:n}$?" Although many solutions will likely be hierarchical combinations of methods, this question is still relevant for each layer independently. Central to answering this question is an understanding of the dynamics of the system at a given level. Currently, the vast majority of opponent modeling approaches are myopic, seeking only to predict the very next opponent state, $O_{t+1}^{1:n}$. In instances where $\Delta t > 1$, search spaces often become very large and the opponent modeling system must resort to sampling. The strategy gradient offers a way to bias samples and drastically reduce the number Monte Carlo rollouts which explore regions of the state space that are unlikely to be realized, similar to pruning techniques conditioned on the estimated strategies of more traditional adversarial search algorithms (Sturtevant & Bowling, 2006; Sturtevant et al., 2006). It may also be possible to perform some form of real-valued landmark identification if an opponent's trajectory through game state space must always pass through the same extrema with respect to some variable.

Estimating gradients requires the underlying value functions, or at the very least, reward functions. Recent advances in inverse reinforcement learning (IRL) (Zhifei & Joo, 2012; Arora & Doshi, 2021) could provide a foundation for techniques seeking to estimate these quantities as a prelude to estimating strategy gradient. Several multi-agent IRL variants been developed, and methods have been tested in both cooperative (Hadfield-Menell et al., 2016; Gaurav & Ziebart, 2019) and adversarial (Tucker et al., 2018; Lin et al., 2017) domains. However, there is substantial work remaining before these techniques are applicable to all the domains in this survey.

The concepts of strategy manifolds and strategy gradients are by no means presented in completion here. Further research will be required to develop these ideas to their full potential, and we believe they offer a promising framework to conceptualize and formalize these problems. Importantly, we also hypothesize that these concepts can be used to learn the rules of a game or the capabilities of different agents given strategies for all agents. Just as young children can both form models of the rules by observing play (Rakoczy et al., 2008), as well as develop strategies given the rules, so too should our agents. A failure to achieve this capability is likely evidence of an incomplete model, or a model that lacks the potential for a general understanding of the situation. Some recent work presents excellent examples of well-designed experiments which verify the efficacy of proposed methods by producing or predicting the type of emergent behavior we might expect from a strategic agent (Tang et al., 2021; Wang et al., 2021).

6.6 Transfer Learning Between Different Rule Sets

Knowledge about strategic principles is almost always transferable between different instances of the same game featuring different opponents. Furthermore, some concepts are also transferable between *different*, related games. For instance, defensive schemes in hockey, soccer, and basketball share a number of similarities such that humans who learn some basic strategic principles in the context of one game can often apply them successfully to the

other games without explicit instruction. Transfer learning (Pan & Yang, 2010; Weiss et al., 2016) allows algorithms to apply knowledge learned in one setting to other, novel settings, and is already an active area of research in multi-agent systems (Da Silva & Costa, 2019) and within adversarial, partially observable games (Shao et al., 2018). However, current opponent modeling research can handle facing a new opponent, but we do not have existing techniques for engaging in a game with different but related rules.

Successful applications of transfer learning to opponent modeling will reduce the time needed to model opponents accurately during engagements, since some portion of the strategy of the game can be transferred before the game begins. There are few examples of this in the opponent modeling literature, but there are some, including Hawasly and Ramamoorthy (2013) who are able to learn sub-policies that are transferable between games in simulated robot soccer. An attractive augmentation to transfer learning is the concept of policy distillation (Rusu et al., 2015), where a policy is represented in a compressed, abstracted form, allowing it to be matched or applied to families of cases. Similarly, research by Wu et al. (2021) learns a base policy and then adapts online to new opponents. The key to their approach is generating a population of opponents with different strategies to train against. It is not hard to imagine policies representing behavior such as "always stay between the opponent and the goal while defending" or "avoid co-locating with teammates while on offense", which we know to be features of high level play, distilled and transferred between contests or even between similar games.

7. Discussion and Conclusion

This survey provides several contributions. We review a large number of existing works and provide a means to easily compare them. We also introduce the concepts and formalism for strategy manifolds and strategy gradients. Last, we discuss future research directions and opportunities for opponent modeling. Here, we would also like to provide some general takeaways for researchers and practitioners interested in doing research on opponent modeling or using opponent modeling systems.

When to Employ Opponent Modeling Opponent modeling is most effective when the system meets some or all of the criteria set out in the definition we use in section 2. It must be possible to observe the adversary, and these observations must provide information about the adversary that can be used to exploit their strategy. This observability constraint may be a limiting factor if the inference or learning framework is not sample efficient. For situations where the optimal strategy is known, opponent modeling may be useful if the opponent deviates from an equilibrium strategy and affords the opportunity to exploit a strategic weakness. When the optimal strategy is not known or not unique, it is far more likely that the adversary being modeled is suboptimal, and it is therefore possible to exploit their strategy in some way. If the adversary is optimal, then their strategy cannot be exploited even if it is known. Lastly, opponent modeling techniques often offer the biggest advantage over game-theoretic techniques when the state space is partially observable. That is, when there are variables that impact the future states of the game, but whose values are not known by all agents. Opponent modeling techniques could also be adapted for use in cooperative domains simply to model the goals or policy of an external agent, where the resultant model could then be used to reduce the likelihood of bad outcomes instead of increase the

likelihood of exploitation. In terms of which general class of learning algorithm to employ, there is one tradeoff that is particularly important. This is the added complexity of defining Z and $h()$ when using discriminative or generative models versus the data inefficiency of policy approximation. We should also note that even approaches which learn a policy for an adversarial domain directly, without explicitly performing opponent modeling, are still constructing an opponent model which is latent within the resultant policy.

Communicating Opponent Modeling Hypotheses Clearly there are many domains where opponent modeling research has already been conducted and it seems the utility of hypotheses regarding novel domains at this point is low. Similarly, many algorithms have been applied to learn opponent models, and while algorithmic improvements are certainly still possible, many of the challenges identified here are not solvable simply by using a new learning or inference method. One of the goals of this survey is to offer concepts and vocabulary which facilitate more fruitful hypotheses. Here, we reiterate several tools we hope researchers will add to their toolbox. First, hierarchical composition of opponent modeling methods seems to be very promising for games like robot soccer where agents may perform actions individually or in a cooperative manner with some or all of their team. Although we highlight several hierarchical systems in this paper, the vast majority of proposed techniques do not take advantage of the natural hierarchy formed by atomic, branch, and complete abstractions. Second, as can be seen by adopting the general expression proposed in section 5, although there are many potential sets of information opponent modeling systems may use, most approaches use only a fraction of the potential information available. In fact, for many systems the expressions are identical. This leads us to believe there is substantial room for innovation in developing systems that can leverage additional sources of data. Last, we have noticed that many papers do not present ablation tests with respect to internal representations of opponents or game states. We propose using the concepts of strategy manifold and strategy gradient to assist in feature validation as well as to promote hypotheses regarding the dynamics of game states.

Recommendations for Improving Experimentation There are two prominent weaknesses in the existing opponent modeling literature with respect to experimental design and validation of proposed methods. Most importantly, the majority of research does not actually test hypotheses related to assumption (5) from section 2, that opponent modeling systems uncover exploitable information about opponent strategies. Instead, the accuracy for an intermediate classification component is often reported. While this is useful in developing better intermediate classification algorithms, it does not by itself inevitably lead to better opponent modeling systems. Ideally, experiments should show that systems with higher accuracy in predicting, say, whether a given robot is an attacker or defender, also have a higher expected score differential. This is not simply a missing data point for which we may make a reasonable guess. This practice obfuscates the challenging problem of taking a class label and deriving a prediction. It also avoids altogether defining and implementing $h()$ or justifying Z, which severely limits the conclusions which can be drawn. The second weakness is the lack of validation of different feature sets and feature extraction techniques. Most papers present only a single method, do not discuss its genesis, and do not attempt to validate the importance or necessity of any of the features.

We also think experiments which incorporate multiple domains and show the opponent modeling system learning fundamental strategic concepts across similar domains will be especially compelling. We should expect opponent modeling systems to predict the same types of emergent strategic behavior and co-evolution of strategies that we observe whenever the rules of a sport are changed, when a competitive video game receives a major patch, or when a new military technology becomes widely available. The only way to benchmark progress towards such goals is to run these experiments.

Acknowledgements

We would like to acknowledge the anonymous reviewers for their incredibly helpful and insightful reviews, without which this paper would not be as clear or compelling. We also thank the editors for their patience and guidance.

References

Adachi, Y., Ito, M., & Naruse, T. (2016). Classifying the strategies of an opponent team based on a sequence of actions in the RoboCup SSL. In *Robot World Cup*, pp. 109–120. Springer.

Ahmadi, M., Lamjiri, A. K., Nevisi, M. M., Habibi, J., & Badie, K. (2003). Using a two-layered case-based reasoning for prediction in soccer coach. In *International Conference on Machine Learning; Models, Technologies and Applications*, pp. 181–185.

Al-Shedivat, M., Bansal, T., Burda, Y., Sutskever, I., Mordatch, I., & Abbeel, P. (2017). Continuous adaptation via meta-learning in nonstationary and competitive environments. *arXiv preprint arXiv:1710.03641*.

Albrecht, S. V., & Stone, P. (2018). Autonomous agents modelling other agents: A comprehensive survey and open problems. *Artificial Intelligence*, *258*, 66–95.

Alimadadi, S., Mesbah, A., & Pattabiraman, K. (2018). Inferring hierarchical motifs from execution traces. In *Proceedings of the 40th International Conference on Software Engineering (ICSE)*.

An, V. C., & Sandholm, T. (2003). AWESOME: A general multiagent learning algorithm that converges in self-play and learns a best response against stationary opponents. In *Proceedings of the Twentieth International Conference on Machine Learning*, pp. 83–90.

Archibald, C., & Nieves-Rivera, D. (2018). Execution skill estimation. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 1859–1861.

Archibald, C., & Nieves-Rivera, D. (2019). Bayesian execution skill estimation. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*, Vol. 33, pp. 6014–6021.

Arora, S., & Doshi, P. (2021). A survey of inverse reinforcement learning: Challenges, methods and progress. *Artificial Intelligence*, 103500.

Baarslag, T., Hendrikx, M. J. C., Hindriks, K. V., & Jonker, C. M. (2016). Learning about the opponent in automated bilateral negotiation: A comprehensive survey of opponent modeling techniques. *Autonomous Agents and Multi-Agent Systems*, *30*(5), 849–898.

Baez, S. (2015). Predicting opponent team activity in a RoboCup environment. *arXiv preprint arXiv:1503.01446*.

Baker, C. L., Tenenbaum, J. B., & Saxe, R. R. (2007). Goal inference as inverse planning. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, Vol. 29.

Bakkes, S. C., Spronck, P. H., & van Lankveld, G. (2012). Player behavioural modelling for video games. *Entertainment Computing*, *3*(3), 71–79.

Ball, D., & Wyeth, G. (2003). Classifying an opponent's behaviour in robot soccer. In *Proceedings of the Australasian Conference on Robotics and Automation*.

Bard, N., & Bowling, M. (2007). Particle filtering for dynamic agent modelling in simplified poker. In *Proceedings of the National Conference on Artificial Intelligence*, Vol. 22, p. 515.

Beetz, M., Hoyningen-Huene, N. v., Bandouch, J., Kirchlechner, B., Gedikli, S., & Maldonado, A. (2006). Camera-based observation of football games for analyzing multi-agent activities. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 42–49.

Beetz, M., Kirchlechner, B., & Lames, M. (2005). Computerized real-time analysis of football games. *IEEE Pervasive Computing*, *4*(3), 33–39.

Belghazi, M. I., Baratin, A., Rajeswar, S., Ozair, S., Bengio, Y., Courville, A., & Hjelm, R. D. (2018). Mine: Mutual information neural estimation. *arXiv preprint arXiv:1801.04062*.

Bengio, Y., & Frasconi, P. (1995). An input output HMM architecture. In *Advances in Neural Information Processing Systems*, pp. 427–434.

Bernstein, D. S., Givan, R., Immerman, N., & Zilberstein, S. (2002). The complexity of decentralized control of markov decision processes. *Mathematics of Operations Research (MOR)*, *27*(4), 819–840.

Bhandari, I., Colet, E., Parker, J., Pines, Z., Pratap, R., & Ramanujam, K. (1997). Advanced scout: Data mining and knowledge discovery in NBA data. *Data Mining and Knowledge Discovery*, *1*(1), 121–125.

Biermann, A. W., & Feldman, J. A. (1972). On the synthesis of finite-state machines from samples of their behavior. *IEEE Transactions on Computers*, *100*(6), 592–597.

Biswas, J., Mendoza, J. P., Zhu, D., Choi, B., Klee, S., & Veloso, M. (2014). Opponent-driven planning and execution for pass, attack, and defense in a multi-robot soccer team. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems*, pp. 493–500.

Blaylock, N., & Allen, J. (2006). Fast hierarchical goal schema recognition. In *Proceedings of the National Conference on Artificial Intelligence*, Vol. 21, p. 796.

Bombini, G., Di Mauro, N., Ferilli, S., & Esposito, F. (2010). Classifying agent behaviour through relational sequential patterns. In *KES International Symposium on Agent and Multi-Agent Systems: Technologies and Applications*, pp. 273–282. Springer.

Bowling, M., & Veloso, M. (2001). Rational and convergent learning in stochastic games. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, Vol. 17, pp. 1021–1026.

Brockbank, E., & Vul, E. (2021). Formalizing opponent modeling with the rock, paper, scissors game. *Games*, *12*(3), 70.

Brown, N., Bakhtin, A., Lerer, A., & Gong, Q. (2020). Combining deep reinforcement learning and search for imperfect-information games. *Proceedings of the Thirty-Fourth Conference on Neural Information Processing Systems*.

Brown, N., & Sandholm, T. (2019). Solving imperfect-information games via discounted regret minimization. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*, Vol. 33, pp. 1829–1836.

Browne, C. B., Powley, E., Whitehouse, D., Lucas, S. M., Cowling, P. I., Rohlfshagen, P., Tavener, S., Perez, D., Samothrakis, S., & Colton, S. (2012). A survey of Monte Carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, *4*(1), 1–43.

Browning, B., Bruce, J., Bowling, M., & Veloso, M. (2005). STP: Skills, Tactics, and Plays for multi-robot control in adversarial environments. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, *219*(1), 33–52.

Bui, H. H., Venkatesh, S., & West, G. (2002). Policy recognition in the abstract hidden Markov model. *Journal of Artificial Intelligence Research*, *17*, 451–499.

Butler, S., & Demiris, Y. (2009). Predicting the movements of robot teams using generative models. In *Distributed Autonomous Robotic Systems*, Vol. 8, pp. 533–542. Springer.

Carmel, D., & Markovitch, S. (1993). Learning models of opponent's strategy game playing. In *Proceedings of the 1993 AAAI Fall Symposium on Games: Planning and Learning*.

Carmel, D., & Markovitch, S. (1996a). Incorporating opponent models into adversary search. In *Proceedings of the Tenth AAAI Conference on Artificial Intelligence*, Vol. 1, pp. 120–125.

Carmel, D., & Markovitch, S. (1996b). Learning models of intelligent agents. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conference*, Vol. 1, pp. 62–67.

Carmel, D., & Markovitch, S. (1996c). Opponent modeling in multi-agent systems. *Adaption and Learning in Multi-agent Systems*, 40–52.

Ceren, R., He, K., Doshi, P., & Banerjee, B. (2021). PALO bounds for reinforcement learning in partially observable stochastic games. *Neurocomputing*, *420*, 36–56.

Chakraborty, D., Agmon, N., & Stone, P. (2013). Targeted opponent modeling of memory-bounded agents. In *Proceedings of the Adaptive Learning Agents Workshop (ALA)*.

Chen, H., Wang, C., Huang, J., & Gong, J. (2020). Efficient use of heuristics for accelerating XCS-based policy learning in Markov games. *arXiv preprint arXiv:2005.12553*.

Chen, S., & Arkin, R. C. (2021). Counter-misdirection in behavior-based multi-robot teams. In *IEEE International Conference on Intelligence and Safety for Robotics (ISR)*.

Cliff, O. M., Lizier, J. T., Wang, X. R., Wang, P., Obst, O., & Prokopenko, M. (2013). Towards quantifying interaction networks in a football match. In *Robot Soccer World Cup*, pp. 1–12. Springer.

Cliff, O. M., Lizier, J. T., Wang, X. R., Wang, P., Obst, O., & Prokopenko, M. (2017). Quantifying long-range interactions and coherent structure in multi-agent dynamics. *Artificial life*, *23*(1), 34–57.

Da Silva, F. L., & Costa, A. H. R. (2019). A survey on transfer learning for multiagent reinforcement learning systems. *Journal of Artificial Intelligence Research*, *64*, 645–703.

Davis, T., Waugh, K., & Bowling, M. (2019). Solving large extensive-form games with strategy constraints. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*, Vol. 33, pp. 1861–1868.

De Weerd, H., Verbrugge, R., & Verheij, B. (2013). How much does it help to know what she knows you know? an agent-based simulation study. *Artificial Intelligence*, *199*, 67–92.

Devaney, M., & Ram, A. (1998). Needles in a haystack: Plan recognition in large spatial domains involving multiple agents. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence and the Tenth Innovative Applications of Artificial Intelligence Conference*, pp. 942–947.

Ding, S., Zhu, H., Jia, W., & Su, C. (2012). A survey on feature extraction for pattern recognition. *Artificial Intelligence Review*, *37*(3), 169–180.

Donkers, H., Uiterwijk, J. W. H. M., & van den Herik, H. J. (2001). Probabilistic opponent-model search. *Information Sciences*, *135*(3-4), 123–149.

Donkers, H. H. L. M. (2003). Nosce hostem: Searching with opponent models..

Doshi, P., & Gmytrasiewicz, P. J. (2009). Monte Carlo sampling methods for approximating interactive POMDPs. *Journal of Artificial Intelligence Research*, *34*, 297–337.

Doucet, A., De Freitas, N., Murphy, K., & Russell, S. (2000). Rao-Blackwellised particle filtering for dynamic Bayesian networks. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, pp. 176–183.

Duda, R. O., Hart, P. E., & Stork, D. G. (2012). *Pattern Classification*. John Wiley & Sons.

Egorov, M., Kochenderfer, M. J., & Uudmae, J. J. (2016). Target surveillance in adversarial environments using POMDPs. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pp. 2473–2479.

Emery-Montemerlo, R., Gordon, G., Schneider, J., & Thrun, S. (2004). Approximate solutions for partially observable stochastic games with common payoffs. In *Proceedings*

of the Third International Joint Conference on Autonomous Agents and Multiagent Systems, Vol. 1, pp. 136–143.

Erdogan, C., & Veloso, M. M. (2011). Action selection via learning behavior patterns in multi-robot domains. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*.

Ernst, M. D., Perkins, J. H., Guo, P. J., McCamant, S., Pacheco, C., Tschantz, M. S., & Xiao, C. (2007). The Daikon system for dynamic detection of likely invariants. *Science of Computer Programming*, *69*(1-3), 35–45.

Esposito, F., Di Mauro, N., Basile, T., & Ferilli, S. (2008). Multi-dimensional relational sequence mining. *Fundamenta Informaticae*, *89*(1), 23–43.

Everett, R., & Roberts, S. (2018). Learning against non-stationary agents with opponent modelling and deep reinforcement learning. In *2018 AAAI Spring Symposium Series*.

Fagan, M., & Cunningham, P. (2003). Case-based plan recognition in computer games. In *International Conference on Case-Based Reasoning*, pp. 161–170. Springer.

Fagundes, M. S., Meneguzzi, F., Bordini, R. H., & Vieira, R. (2014). Dealing with ambiguity in plan recognition under time constraints. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems*, pp. 389–396.

Fard, A. M., Salmani, V., Naghibzadeh, M., Nejad, S. K., & Ahmadi, H. (2007). Game theory-based data mining technique for strategy making of a soccer simulation coach agent. In *International Symposium on Intelligent Systems and Technologies Applications*, Vol. 2007, pp. 54–65.

Farina, G., Kroer, C., Brown, N., & Sandholm, T. (2019). Stable-predictive optimistic counterfactual regret minimization. In *Proceedings of the International Conference on Machine Learning*, pp. 1853–1862.

Farouk, G. M., Moawad, I. F., & Aref, M. M. (2017). A machine learning based system for mostly automating opponent modeling in real-time strategy games. In *Proceedings of the 12th International Conference on Computer Engineering and Systems (ICCES)*, pp. 337–346. IEEE.

Fischler, M. A., & Bolles, R. C. (1981). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the Association for Computing Machinery*, *24*(6), 381–395.

Floyd, M. W., Karneeb, J., & Aha, D. W. (2017). Case-based team recognition using learned opponent models. In *International Conference on Case-Based Reasoning*, pp. 123–138. Springer.

Frank, E., & Witten, I. H. (1998). Generating accurate rule sets without global optimization. In *Proceedings of the Fifteenth International Conference on Machine Learning*.

Fukushima, T., Nakashima, T., & Akiyama, H. (2017). Online opponent formation identification based on position information. *RoboCup 2017*.

Gallego, V., Naveiro, R., & Insua, D. R. (2019). Reinforcement learning under threats. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*, Vol. 33, pp. 9939–9940.

Ganzfried, S., Nowak, A., & Pinales, J. (2018). Successful Nash equilibrium agent for a 3-player imperfect-information game. *arXiv preprint arXiv:1804.04789*.

Ganzfried, S., & Sandholm, T. (2011). Game theory-based opponent modeling in large imperfect-information games. In *The 10th International Conference on Autonomous Agents and Multiagent Systems*, Vol. 2, pp. 533–540.

Ganzfried, S., & Sandholm, T. (2015). Safe opponent exploitation. *ACM Transactions on Economics and Computation (TEAC)*, *3*(2), 1–28.

Gaurav, S., & Ziebart, B. D. (2019). Discriminatively learning inverse optimal control models for predicting human intentions. In *International Conference on Autonomous Agents and Multiagent Systems*.

Geib, C. W., & Goldman, R. P. (2009). A probabilistic plan recognition algorithm based on plan tree grammars. *Artificial Intelligence*, *173*(11), 1101–1132.

Gold, E. M. (1978). Complexity of automaton identification from given data. *Information and Control*, *37*(3), 302–320.

Gold, K. (2010). Training goal recognition online from low-level inputs in an action-adventure game. In *Sixth Artificial Intelligence and Interactive Digital Entertainment Conference*.

Goldman, A. I., et al. (2012). Theory of mind. *The Oxford Handbook of Philosophy of Cognitive Science*, *1*.

Green, C. (2004). Phased searching with NEAT: Alternating between complexification and simplification. *Unpublished manuscript*.

Grover, A., Al-Shedivat, M., Gupta, J. K., Burda, Y., & Edwards, H. (2018). Learning policy representations in multiagent systems. *arXiv preprint arXiv:1806.06464*.

Hadfield-Menell, D., Russell, S. J., Abbeel, P., & Dragan, A. (2016). Cooperative inverse reinforcement learning. *Advances in Neural Information Processing Systems*, *29*, 3909–3917.

Han, K., Veloso, M., et al. (2000). Automated robot behavior recognition. In *Robotics Research International Symposium*, Vol. 9, pp. 249–256.

Hansen, E. A., Bernstein, D. S., & Zilberstein, S. (2004). Dynamic programming for partially observable stochastic games. In *Proceedings of the Eighteenth AAAI Conference on Artificial Intelligence*, Vol. 4, pp. 709–715.

Hawasly, M., & Ramamoorthy, S. (2013). Lifelong transfer learning with an option hierarchy. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1341–1346. IEEE.

Hayes, R., & Beling, P. (2018). Unsupervised hierarchical clustering of build orders in a real-time strategy game. *The Computer Games Journal*, *7*(1), 5–26.

He, H., Boyd-Graber, J., Kwok, K., & Daumé III, H. (2016). Opponent modeling in deep reinforcement learning. In *International Conference on Machine Learning*, pp. 1804–1813.

Hernandez-Leal, P., Kaisers, M., Baarslag, T., & de Cote, E. M. (2017a). A survey of learning in multiagent environments: Dealing with non-stationarity. *arXiv preprint arXiv:1707.09183*.

Hernandez-Leal, P., Zhan, Y., Taylor, M. E., Sucar, L. E., & de Cote, E. M. (2017b). Efficiently detecting switches against non-stationary opponents. *Autonomous Agents and Multi-Agent Systems*, *31*(4), 767–789.

Heule, M. J., & Verwer, S. (2013). Software model synthesis using satisfiability solvers. *Empirical Software Engineering*, *18*(4), 825–856.

Hoang, T. N., & Low, K. H. (2013). Interactive pomdp lite: Towards practical planning to predict and exploit intentions for interacting with self-interested agents. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*.

Hoang, T. N., Xiao, Y., Sivakumar, K., Amato, C., & How, J. (2017). Near-optimal adversarial policy switching for decentralized asynchronous multi-agent systems. *arXiv preprint arXiv:1710.06525*.

Hoffmann, J., Porteous, J., & Sebastia, L. (2004). Ordered landmarks in planning. *Journal of Artificial Intelligence Research*, *22*, 215–278.

Hong, J. (2001). Goal recognition through goal graph analysis. *Journal of Artificial Intelligence Research*, *15*, 1–30.

Hong, Z.-W., Su, S.-Y., Shann, T.-Y., Chang, Y.-H., & Lee, C.-Y. (2017). A deep policy inference Q-network for multi-agent systems. *arXiv preprint arXiv:1712.07893*.

Howe, A. E., & Cohen, P. R. (1995). Understanding planner behavior. *Artificial Intelligence*, *76*(1-2), 125–166.

Hsieh, J.-L., & Sun, C.-T. (2008). Building a player strategy model by analyzing replays of real-time strategy games. In *Proceedings of the 2008 IEEE International Joint Conference on Neural Networks*, pp. 3106–3111. IEEE.

Huo, X., Ni, X. S., & Smith, A. K. (2007). A survey of manifold-based learning methods. *Recent Advances in Data Mining of Enterprise Data*, 691–745.

Iglesias, J. A., Ledezma, A., & Sanchis, A. (2009). Caos Coach 2006 Simulation Team: An opponent modelling approach. *Computing and Informatics*.

Intille, S. S., & Bobick, A. F. (1999). A framework for recognizing multi-agent action from visual evidence. *Proceedings of the Sixteenth National Conference on Artificial Intelligence and the Eleventh Innovative Applications of Artificial Intelligence Conference*, *99*(518–525).

Intille, S. S., & Bobick, A. F. (2001). Recognizing planned, multiperson action. *Computer Vision and Image Understanding*, *81*(3), 414–445.

Isaacs, R. (1955). A card game with bluffing. *The American Mathematical Monthly*, *62*(2), 99–108.

Izenman, A. J. (2012). Introduction to manifold learning. *Wiley Interdisciplinary Reviews: Computational Statistics*, *4*(5), 439–446.

Johanson, M., & Bowling, M. (2009). Data biased robust counter strategies. In *Artificial Intelligence and Statistics*, pp. 264–271.

Johanson, M., Bowling, M., & Zinkevich, M. (2008). Computing robust counter-strategies. In *In Advances in Neural Information Processing Systems*.

Kabanza, F., Bellefeuille, P., Bisson, F., Benaskeur, A. R., & Irandoust, H. (2010). Opponent behaviour recognition for real-time strategy games. *Plan, Activity, and Intent Recognition*, *10*(5).

Kaminka, G. A., Fidanboylu, M., Chang, A., & Veloso, M. M. (2002). Learning the sequential coordinated behavior of teams from observations. In *Robot Soccer World Cup*, pp. 111–125. Springer.

Kamrani, F., Luotsinen, L. J., & Løvlid, R. A. (2016). Learning objective agent behavior using a data-driven modeling approach. In *2016 IEEE International Conference on Systems, Man, and Cybernetics*, pp. 002175–002181. IEEE.

Kar, D., Ford, B., Gholami, S., Fang, F., Plumptre, A., Tambe, M., Driciru, M., Wanyama, F., Rwetsiba, A., Nsubaga, M., et al. (2017). Cloudy with a chance of poaching: Adversary behavior modeling and forecasting with real-world poaching data. In *Proceedings of the Sixteenth Conference on Autonomous Agents and MultiAgent Systems*, pp. 159–167.

Keren, S., Gal, A., & Karpas, E. (2015). Goal recognition design for non-optimal agents.. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pp. 3298–3304.

Kim, M., & Kim, K. (2017). Opponent modeling based on action table for MCTS-based fighting game AI. In *Proceedings of the IEEE Conference on Computational Intelligence and Games*, pp. 178–180.

Kitano, H., Asada, M., Kuniyoshi, Y., Noda, I., & Osawa, E. (1997). RoboCup: The robot world cup initiative. In *Proceedings of the First International Conference on Autonomous Agents*, AGENTS '97, pp. 340–347, New York, NY, USA. ACM.

Kovalchik, S., Ingram, M., Weeratunga, K., & Goncu, C. (2020). Space-time VON CRAMM: Evaluating decision-making in tennis with Variational generatiON of Complete Resolution Arcs via Mixture Modeling. *arXiv preprint arXiv:2005.12853*.

Krka, I., Brun, Y., & Medvidovic, N. (2014). Automatic mining of specifications from invocation traces and method invariants. In *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pp. 178–189. Association for Computing Machinery.

Kroer, C., & Sandholm, T. (2020). Limited lookahead in imperfect-information games. *Artificial Intelligence*, 103218.

Kuhlmann, G., Knox, W. B., & Stone, P. (2006). Know thine enemy: A champion RoboCup coach agent. In *Proceedings of the National Conference on Artificial Intelligence*, Vol. 21, p. 1463–1468.

Lattner, A. D., Miene, A., Visser, U., & Herzog, O. (2005). Sequential pattern mining for situation and behavior prediction in simulated robotic soccer. In *Robot Soccer World Cup*, pp. 118–129. Springer.

Laviers, K., & Sukthankar, G. (2011). A real-time opponent modeling system for Rush Football. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, Vol. 22, p. 2476–2481.

Laviers, K., Sukthankar, G., Aha, D. W., Molineaux, M., Darken, C., et al. (2009). Improving offensive performance through opponent modeling. In *Proceedings of the AAAI Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE)*.

Ledezma, A., Aler, R., Sanchis, A., & Borrajo, D. (2009). OMBO: An opponent modeling approach. *AI Communications*, *22*(1), 21–35.

Leece, M., & Jhala, A. (2014). Opponent state modeling in RTS games with limited information using Markov random fields. In *2014 IEEE Conference on Computational Intelligence and Games*, pp. 1–7. IEEE.

Li, R., & Chellappa, R. (2010). Group motion segmentation using a spatio-temporal driving force model. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2038–2045. IEEE.

Li, R., Chellappa, R., & Zhou, S. K. (2009). Learning multi-modal densities on discriminative temporal interaction manifold for group activity recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2450–2457. IEEE.

Li, W., Wang, X., Jin, B., Sheng, J., & Zha, H. (2021). Dealing with non-stationarity in multi-agent reinforcement learning via trust region decomposition. *arXiv preprint arXiv:2102.10616*.

Lin, X., Beling, P. A., & Cogill, R. (2017). Multiagent inverse reinforcement learning for two-person zero-sum games. *IEEE Transactions on Games*, *10*(1), 56–68.

Liu, A., Chen, J., Yu, M., Zhai, Y., Zhou, X., & Liu, J. (2018). Watch the unobserved: A simple approach to parallelizing Monte Carlo tree search. *arXiv preprint arXiv:1810.11755*.

Lockett, A. J., Chen, C. L., & Miikkulainen, R. (2007). Evolving explicit opponent models in game playing. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*, pp. 2106–2113. Association for Computing Machinery.

Lu, F., Yamamoto, K., Nomura, L. H., Mizuno, S., Lee, Y., & Thawonmas, R. (2013). Fighting game artificial intelligence competition platform. In *Proceedings of the 2nd IEEE Global Conference on Consumer Electronics*, pp. 320–323. IEEE.

Lucey, P., Oliver, D., Carr, P., Roth, J., & Matthews, I. (2013). Assessing team strategy using spatiotemporal data. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1366–1374.

Marín, C. A., Castillo, L. P., & Garrido, L. (2005). Dynamic adaptive opponent modeling: Predicting opponent motion while playing soccer. In *Fifth European Workshop on Adaptive Agents and Multiagent Systems, Paris, France*.

Markovitch, S., & Reger, R. (2005). Learning and exploiting relative weaknesses of opponent agents. *Autonomous Agents and Multi-Agent Systems*, *10*(2), 103–130.

Martin, B. (1995). Instance-based learning: Nearest neighbour with generalisation. *University of Waikato, Department of Computer Science*.

Masters, P., & Sardina, S. (2017). Cost-based goal recognition for path-planning. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, pp. 750–758.

Masters, P., & Sardina, S. (2021). Expecting the unexpected: Goal recognition for rational and irrational agents. *Artificial Intelligence*, 103490.

McCracken, P., & Bowling, M. (2004). Safe strategies for agent modelling in games. In *Proceedings of the AAAI Fall Symposium on Artificial Multi-agent Learning*, pp. 103–110.

Meek, C., & Glymour, C. (1994). Conditioning and intervening. *The British journal for the philosophy of science*, *45*(4), 1001–1021.

Mescheder, D., Tuyls, K., & Kaisers, M. (2011). Opponent modeling with POMDPs. In *Proceedings of the 23rd Belgium-Netherlands Conference on Artificial Intelligence (BNAIC 2011)*, pp. 152–159.

Molineaux, M., Aha, D. W., & Sukthankar, G. (2009). Beating the defense: Using plan recognition to inform learning agents. Tech. rep., DTIC Document.

Mordatch, I., & Abbeel, P. (2018). Emergence of grounded compositional language in multi-agent populations. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*.

Müllner, D. (2011). Modern hierarchical agglomerative clustering algorithms. *arXiv preprint arXiv:1109.2378*.

Nair, R., Tambe, M., Marsella, S., & Raines, T. (2004). Automated assistants for analyzing team behaviors. *Autonomous Agents and Multi-Agent Systems*, *8*(1), 69–111.

Ng, B., Boakye, K., Meyers, C., & Wang, A. (2012). Bayes-adaptive interactive POMDPs. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*.

Ng, B., Meyers, C., Boakye, K., & Nitao, J. (2010). Towards applying interactive POMDPs to real-world adversary modeling. In *The Twenty-Second Annual Conference on Innovative Applications of Artificial Intelligence (IAAI)*.

Nguyen, T. H., Yang, R., Azaria, A., Kraus, S., & Tambe, M. (2013). Analyzing the effectiveness of adversary modeling in security games. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*.

Ontanón, S., Synnaeve, G., Uriarte, A., Richoux, F., Churchill, D., & Preuss, M. (2013). A survey of real-time strategy game AI research and competition in StarCraft. *IEEE Transactions on Computational Intelligence and AI in Games*, *5*(4), 293–311.

Pan, S. J., & Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, *22*(10), 1345–1359.

Panella, A., & Gmytrasiewicz, P. (2017). Interactive POMDPs with finite-state models of other agents. *Autonomous Agents and Multi-Agent Systems*, *31*(4), 861–904.

Papoudakis, G., & Albrecht, S. V. (2020). Variational autoencoders for opponent modeling in multi-agent systems. *arXiv preprint arXiv:2001.10829*.

Papoudakis, G., Christianos, F., Rahman, A., & Albrecht, S. V. (2019). Dealing with non-stationarity in multi-agent deep reinforcement learning. *arXiv preprint arXiv:1906.04737*.

Pereira, R. F., Oren, N., & Meneguzzi, F. (2017). Landmark-based heuristics for goal recognition. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*.

Perše, M., Kristan, M., Kovačič, S., Vučkovič, G., & Perš, J. (2009). A trajectory-based analysis of coordinated team activity in a basketball game. *Computer Vision and Image Understanding*, *113*(5), 612–621.

Pita, J., Jain, M., Marecki, J., Ordóñez, F., Portway, C., Tambe, M., Western, C., Paruchuri, P., & Kraus, S. (2008). Deployed ARMOR protection: The application of a game theoretic model for security at the Los Angeles International Airport. In *Proceedings of the Seventh International Joint Conference on Autonomous Agents and Multiagent Systems: Industrial Track*, pp. 125–132.

Pitt, L., & Warmuth, M. K. (1993). The minimum consistent DFA problem cannot be approximated within any polynomial. *Journal of the ACM (JACM)*, *40*(1), 95–142.

Pless, R., & Souvenir, R. (2009). A survey of manifold learning for images. *IPSJ Transactions on Computer Vision and Applications*, *1*, 83–94.

Ponsen, M., De Jong, S., & Lanctot, M. (2011). Computing approximate Nash equilibria and robust best-responses using sampling. *Journal of Artificial Intelligence Research*, *42*, 575–605.

Pourmehr, S., & Dadkhah, C. (2011). An overview on opponent modeling in robocup soccer simulation 2D. In *Robot Soccer World Cup*, pp. 402–414. Springer.

Powers, R., & Shoham, Y. (2005). Learning against opponents with bounded memory. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, Vol. 5, pp. 817–822.

Pozanco, A., Yolanda, E., Fernández, S., & Borrajo, D. (2018). Counterplanning using goal recognition and landmarks. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, pp. 4808–4814.

Premack, D., & Woodruff, G. (1978). Does the chimpanzee have a theory of mind?. *Behavioral and Brain Sciences*, *1*(4), 515–526.

Quinlan, J. R. (1993). Combining instance-based and model-based learning. In *Proceedings of the Tenth International Conference on Machine Learning*, pp. 236–243.

Rahman, M., & Oh, J. C. (2018). Online learning for patrolling robots against active adversarial attackers. In *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, pp. 477–488. Springer.

Rakoczy, H., Warneken, F., & Tomasello, M. (2008). The sources of normativity: Young children's awareness of the normative structure of games. *Developmental Psychology*, *44*(3), 875.

Ramırez, M., & Geffner, H. (2009). Plan recognition as planning. In *Proceedings of the Twenty-First International Joint Conference on Artifical Intelligence*, pp. 1778–1783. Morgan Kaufmann Publishers Inc.

Ramırez, M., & Geffner, H. (2010). Probabilistic plan recognition using off-the-shelf classical planners. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, pp. 1121–1126.

Ramırez, M., & Geffner, H. (2011). Goal recognition over POMDPs: Inferring the intention of a POMDP agent. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, pp. 2009–2014.

Rathnasabapathy, B., Doshi, P., & Gmytrasiewicz, P. (2006). Exact solutions of interactive POMDPs using behavioral equivalence. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 1025–1032.

Riley, P., & Veloso, M. (2000). On behavior classification in adversarial environments. In *Distributed Autonomous Robotic Systems*, Vol. 4, pp. 371–380. Springer.

Riley, P., & Veloso, M. (2001). Recognizing probabilistic opponent movement models. In *Robot Soccer World Cup*, pp. 453–458. Springer.

Rosman, B., Hawasly, M., & Ramamoorthy, S. (2016). Bayesian policy reuse. *Machine Learning*, *104*(1), 99–127.

Rovatsos, M., Weiß, G., & Wolf, M. (2003). Multiagent learning for open systems: A study in opponent classification. In *Adaptive Agents and Multi-agent Systems*, pp. 66–87. Springer.

Rusu, A. A., Colmenarejo, S. G., Gulcehre, C., Desjardins, G., Kirkpatrick, J., Pascanu, R., Mnih, V., Kavukcuoglu, K., & Hadsell, R. (2015). Policy distillation. *arXiv preprint arXiv:1511.06295*.

Sadilek, A., & Kautz, H. A. (2010). Recognizing multi-agent activities from GPS data.. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, Vol. 39, p. 109.

Sandholm, T. (2007). Perspectives on multiagent learning. *Artificial Intelligence*, *171*(7), 382–391.

Saria, S., & Mahadevan, S. (2004). Probabilistic plan recognition in multiagent systems. In *International Conference on Automated Planning and Scheduling*, pp. 287–296.

Schadd, F., Bakkes, S., & Spronck, P. (2007). Opponent modeling in real-time strategy games. In *GAMEON*, pp. 61–70.

Seuken, S., & Zilberstein, S. (2008). Formal models and algorithms for decentralized decision making under uncertainty. *Autonomous Agents and Multi-Agent Systems (JAAMAS)*, *17*(2), 190–250.

Shao, K., Zhu, Y., & Zhao, D. (2018). StarCraft micromanagement with reinforcement learning and curriculum transfer learning. *IEEE Transactions on Emerging Topics in Computational Intelligence*, *3*(1), 73–84.

Sheikhpour, R., Sarram, M. A., Gharaghani, S., & Chahooki, M. A. Z. (2017). A survey on semi-supervised feature selection methods. *Pattern Recognition*, *64*, 141–158.

Shen, M., & How, J. P. (2019a). Active perception in adversarial scenarios using maximum entropy deep reinforcement learning. *arXiv preprint arXiv:1902.05644*.

Shen, M., & How, J. P. (2019b). Robust opponent modeling via adversarial ensemble reinforcement learning in asymmetric imperfect-information games. *arXiv preprint arXiv:1909.08735*.

Shieh, E., An, B., Yang, R., Tambe, M., Baldwin, C., DiRenzo, J., Maule, B., & Meyer, G. (2012). Protect: A deployed game theoretic system to protect the ports of the United States. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems*, Vol. 1, pp. 13–20.

Siddiquie, B., Yacoob, Y., & Davis, L. (2009). Recognizing plays in American Football videos. *University of Maryland*.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016). Mastering the game of go with deep neural networks and tree search. *nature*, *529*(7587), 484–489.

Sonu, E., & Doshi, P. (2015). Scalable solutions of interactive POMDPs using generalized and bounded policy iteration. *Autonomous Agents and Multi-Agent Systems*, *29*(3), 455–494.

Spronck, P., & den Teuling, F. (2010). Player modeling in Civilization IV. In *Sixth Artificial Intelligence and Interactive Digital Entertainment Conference*.

Stankiewicz, J. A., & Schadd, M. P. (2009). Opponent modeling in Stratego. *Natural Computing*.

Stanley, K. O., & Miikkulainen, R. (2002). Continual coevolution through complexification. In *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation*, pp. 113–120.

Steffens, T. (2002). Feature-based declarative opponent-modelling in multi-agent systems. *Robot Soccer World Cup*.

Steffens, T. (2005). Similarity-based opponent modelling using imperfect domain theories. In *IEEE Conference on Computational Inteligence in Games (CIG)*.

Stone, P., Riley, P., & Veloso, M. (2000). Defining and using ideal teammate and opponent agent models: A case study in robotic soccer. In *Fourth International Conference on MultiAgent Systems*, pp. 441–442. IEEE.

Stone, P., & Veloso, M. (2000). Layered learning. In *European Conference on Machine Learning*, pp. 369–381. Springer.

Storcheus, D., Rostamizadeh, A., & Kumar, S. (2015). A survey of modern questions and challenges in feature extraction. In *Feature Extraction: Modern Questions and Challenges*, pp. 1–18.

Sturtevant, N. (2004). Current challenges in multi-player game search. In *International Conference on Computers and Games*, pp. 285–300. Springer.

Sturtevant, N., & Bowling, M. (2006). Robust game play against unknown opponents. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 713–719.

Sturtevant, N., Zinkevich, M., & Bowling, M. (2006). Prob-max^n: Playing N-player games with opponent models. In *Proceedings of the Twentieth AAAI Conference on Artificial Intelligence*, Vol. 6, pp. 1057–1063.

Sukthankar, G., Geib, C., Bui, H. H., Pynadath, D., & Goldman, R. P. (2014). *Plan, Activity, and Intent Recognition: Theory and Practice*. Newnes.

Sukthankar, G., & Sycara, K. (2006). Robust recognition of physical team behaviors using spatio-temporal models. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 638–645.

Sukthankar, G., & Sycara, K. (2007). Policy recognition for multi-player tactical scenarios. In *Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multiagent Systems*, p. 16.

Synnaeve, G., & Bessiere, P. (2011). A Bayesian model for opening prediction in RTS games with application to StarCraft. In *IEEE Conference on Computational Intelligence and Games (CIG)*, pp. 281–288. IEEE.

Synnaeve, G., Lin, Z., Gehring, J., Gant, D., Mella, V., Khalidov, V., Carion, N., & Usunier, N. (2018). Forward modeling for partial observation strategy games: A StarCraft defogger. In *Advances in Neural Information Processing Systems*, pp. 10738–10748.

Takács, B., Butler, S., & Demiris, Y. (2007). Multi-agent behaviour segmentation via spectral clustering. In *Proceedings of the PAIR Workshop at the Twenty-First AAAI Conference on Artificial Intelligence*, pp. 74–81.

Tang, Z., Yu, C., Chen, B., Xu, H., Wang, X., Fang, F., Du, S., Wang, Y., & Wu, Y. (2021). Discovering diverse multi-agent strategic behavior via reward randomization. *arXiv preprint arXiv:2103.04564*.

Tang, Z., Zhu, Y., Zhao, D., & Lucas, S. M. (2020). Enhanced rolling horizon evolution algorithm with opponent model learning: Results for the fighting game AI competition. *arXiv preprint arXiv:2003.13949*.

Tian, R., Sun, L., & Tomizuka, M. (2021a). Bounded risk-sensitive Markov games: Forward policy design and inverse reward learning with iterative reasoning and cumulative prospect theory. In *Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence*.

Tian, R., Tomizuka, M., & Sun, L. (2021b). Learning human rewards by inferring their latent intelligence levels in multi-agent games: A theory-of-mind approach with application to driving data. *arXiv preprint arXiv:2103.04289*.

Torkaman, A., & Safabakhsh, R. (2019). Robust opponent modeling in real-time strategy games using Bayesian Networks. *Journal of AI and Data Mining*, *7*(1), 149–159.

Trevizan, F. W., & Veloso, M. M. (2010). Learning opponent's strategies in the RoboCup Small Size League. In *Proceedings of the Ninth International Conference on Autonomous Agents and Multiagent Systems*, Vol. 10.

Tucker, A., Gleave, A., & Russell, S. (2018). Inverse reinforcement learning for video games. *arXiv preprint arXiv:1810.10593*.

Vail, D. L., & Veloso, M. M. (2008). Feature selection for activity recognition in multi-robot domains.. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence*, Vol. 8, pp. 1415–1420.

Vail, D. L., Veloso, M. M., & Lafferty, J. D. (2007). Conditional random fields for activity recognition. In *Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multiagent Systems*, p. 235.

Vered, M., & Kaminka, G. A. (2017). Heuristic online goal recognition in continuous domains. *arXiv preprint arXiv:1709.09839*.

Visser, U., & Weland, H.-G. (2002). Using online learning to analyze the opponent's behavior. In *Robot Soccer World Cup*, pp. 78–93. Springer.

Von Neumann, J., & Morgenstern, O. (1944). *Theory of Games and Economic Behavior*. Princeton University Press.

Wang, J., Zhu, T., Li, H., Hsueh, C.-H., & Wu, I.-C. (2017). Belief-state Monte Carlo tree search for Phantom Go. *IEEE Transactions on Games*, *10*(2), 139–154.

Wang, M., Wang, Z., Talbot, J., Gerdes, J. C., & Schwager, M. (2021). Game-theoretic planning for self-driving cars in multivehicle competitive scenarios. *IEEE Transactions on Robotics*.

Wang, Z., Boularias, A., Mülling, K., & Peters, J. (2011). Balancing safety and exploitability in opponent modeling. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*.

Weber, B. G., & Mateas, M. (2009). A data mining approach to strategy prediction. In *2009 IEEE Symposium on Computational Intelligence and Games*, pp. 140–147. IEEE.

Wei, X., Lucey, P., Morgan, S., & Sridharan, S. (2013). Sweet-spot: Using spatiotemporal data to discover and predict shots in tennis. In *MIT Sloan Sports Analytics Conference*.

Weiss, K., Khoshgoftaar, T. M., & Wang, D. (2016). A survey of transfer learning. *Journal of Big data*, *3*(1), 9.

Wendler, J., & Bach, J. (2003). Recognizing and predicting agent behavior with case based reasoning. In *Robot Soccer World Cup*, pp. 729–738. Springer.

Willmott, S., Richardson, J., Bundy, A., & Levine, J. (2001). Applying adversarial planning techniques to Go. *Theoretical Computer Science*, *252*(1-2), 45–82.

Wu, Z., Li, K., Zhao, E., Xu, H., Zhang, M., Fu, H., An, B., & Xing, J. (2021). L2E: Learning to exploit your opponent. *arXiv preprint arXiv:2102.09381*.

Wunder, M., Kaisers, M., Yaros, J. R., & Littman, M. (2011). Using iterated reasoning to predict opponent strategies. In *The 10th International Conference on Autonomous Agents and Multiagent Systems*, Vol. 2, pp. 593–600.

Xu, Z., & Julius, A. A. (2018). Census signal temporal logic inference for multiagent group behavior analysis. *IEEE Transactions on Automation Science and Engineering*, *15*(1), 264–277.

Yang, R., Ford, B., Tambe, M., & Lemieux, A. (2014). Adaptive resource allocation for wildlife protection against illegal poachers. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems*, pp. 453–460.

Yang, T., Hao, J., Meng, Z., Zhang, C., Zheng, Y., & Zheng, Z. (2019). Towards efficient detection and optimal response against sophisticated opponents. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pp. 623–629.

Yasui, K., Kobayashi, K., Murakami, K., & Naruse, T. (2013). Analyzing and learning an opponent's strategies in the RoboCup Small Size League. In *Robot Soccer World Cup*, pp. 159–170. Springer.

Yin, Q., Yue, S., Zha, Y., & Jiao, P. (2016). A semi-Markov decision model for recognizing the destination of a maneuvering agent in real time strategy games. *Mathematical Problems in Engineering*, *2016*.

Yolanda, E., R-Moreno, M. D., Smith, D. E., et al. (2015). A fast goal recognition technique based on interaction estimates. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Yu, X., Jiang, J., Jiang, H., & Lu, Z. (2021). Model-based opponent modeling. *arXiv preprint arXiv:2108.01843*.

Zeng, Y., & Doshi, P. (2012). Exploiting model equivalences for solving interactive dynamic influence diagrams. *Journal of Artificial Intelligence Research*, *43*, 211–255.

Zhang, Y., Rădulescu, R., Mannion, P., Roijers, D. M., & Nowé, A. (2020). Opponent modelling using policy reconstruction for multi-objective normal form games. In *Proceedings of the Adaptive and Learning Agents Workshop (ALA-20) at AAMAS.(under review)*.

Zheng, Y., Meng, Z., Hao, J., Zhang, Z., Yang, T., & Fan, C. (2018). A deep bayesian policy reuse approach against non-stationary agents. In *Advances in Neural Information Processing Systems*, pp. 954–964.

Zhifei, S., & Joo, E. M. (2012). A survey of inverse reinforcement learning techniques. *International Journal of Intelligent Computing and Cybernetics*.

Zhu, X. J. (2005). Semi-supervised learning literature survey. Tech. rep., University of Wisconsin-Madison Department of Computer Sciences.

Zhuo, H. H., Yang, Q., & Kambhampati, S. (2012). Action-model based multi-agent plan recognition. In *Advances in Neural Information Processing Systems*, pp. 368–376.

Zhuo, H. H., & Li, L. (2011). Multi-agent plan recognition with partial team traces and plan libraries. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*.