Weighted Prediction Divergence for Metareasoning

Brett Borghetti and **Maria Gini**
Dept of Computer Science and Engineering
University of Minnesota,
200 Union St SE, Minneapolis, MN 55455
borg@cs.umn.edu and gini@cs.umn.edu
http://www.cs.umn.edu/~borg

Abstract

One of the most important elements of agent performance in multi-agent systems is the ability for an agent to predict how other agents will behave. In many domains there are often different modeling systems already available that one could use to make behavior predictions, but the choice of the best one for a particular domain and a specific set of agents is often unclear. To find the best available prediction, we would like to know which model would perform best in each possible world state of the domain. However, when we have limited resources and each prediction query has a cost we may need to decide which queries to pursue using only estimates of their benefit and cost: metareasoning. To estimate the benefit of the computation, a metareasoner needs a robust measurement of performance quality. In this work we present a metareasoning system that relies on a prediction performance measurement, and we propose a novel model performance measurement that fulfils this need: Weighted Prediction Divergence.

Introduction

Agent models are internal representations of other agents in an environment. Agents with models of others can play out "what-if" scenarios to determine potential responses of target agents to each of the modeling agent's possible actions. This type of reasoning can help the agent choose the best course of action.

Agent modeling separates the characteristics of an agent from its environment. When an agent models another, the modeler's goal is to either *predict* future behavior or *explain* past behavior by identifying the beliefs of the modeled agent. In this work, we focus specifically on measuring the ability of models to make predictions, in an attempt to answer the following important metareasoning[1] question: Given a world state and a collection of candidate models, how can we a-priori select the one which is most likely to perform the best in that state? When computational cost is an issue, the answer to this question lies within the answers to several related questions:

- How do we automatically generate a contextual abstraction[2] of the world state space?
- How can we estimate a model's general prediction quality in a specific context without measuring quality at every possible world state within that context?
- How do we compare the relative performance of several heterogenous models' predictive capabilities in a specific context?

While the ability to characterize relative performance of heterogenous models is important for a wide collection of metareasoning strategies, and is thus the main thrust of this paper, we must first examine a specific metareasoning strategy such that the desiderata for the relative performance measurement can be derived. In the next section, we propose a metareasoning system that could be coupled with a robust measurement of model prediction performance to yield a powerful new method of model selection for an agent. After reviewing some of the existing methods used for performance measurement, we present the main contribution of the paper - a new method for characterizing model prediction performance. We then introduce one target domain for our empirical testing of this prediction performance measurement. We discuss empirical results using this method and conclude with a description of future work required to realize the overall metareasoning system.

Context-aware Metareasoning for Prediction-Model Selection

Consider an agent trying to find high-value actions to take in some world where the value of the action is partially dependent on the behavior of the other agents in the world. We describe the behaviors of the agents (and the way those behaviors affect and are affected by the world) using an extensive-form game tree where nodes describe world states and edges describe actions taken by specific agents. The value of a world state can be estimated by summing the values of its branches, weighted by the likelihood of the branch (action) being chosen. This process can be used at each sub-branch to find the value of the sub-branches recursively. The recursion bottoms out when a branch is terminated at a leaf node that contains a real-world value. Each node in the tree may also have an intrinsic value which can be included in the calculation.

[1] See (Cox & Raja 2007) for a primer on metareasoning.

[2] For this work, we define a *context* to be a collection of world states. Forming a set of contexts from the domain's world states requires a method of abstraction.

The well-known minimax algorithm discussed in (Russell & Norvig 2003) describes this process when there are two agents and the game is zero-sum (in every outcome one player gains exactly the value the other player loses) and the agents are assumed to have perfect (not bounded) rationality. Many extensions to minimax such as (Carmel & Markovitch 1996a; Luckhardt & Irani 1986; Stone, Riley, & Veloso 2000; Sturtevant, Zinkevich, & Bowling 2006) have been developed which allow the assumption of perfect rationality to be relaxed. By replacing the minimizer with an algorithm that predicts the opponent's behavior at a given node as in (Carmel & Markovitch 1996a) we obtain an algorithm which selects actions with the highest value when playing against a specific modeled opponent. When the model is estimating the other agent's probability distribution over actions, then the quality of the value calculation is dependent upon how similar the predicted behavior distribution is to the actual behavior distribution.

If we want to have a metareasoner that makes decisions about which calculations to carry out using the utility of a calculation as a criteria, we need to know the cost of the calculation and the value of the calculation (Russell & Wefald 1991). In the setting described above, the relative value of several candidate calculations (queries of different predictive models) can be estimated by the quality of their predictions at the specified world state. Unfortunately, measuring the predictive quality of every model at every possible world state in a branch of the game tree is at least as expensive as obtaining all the predictions in the first place - thus computing prediction quality for every node in a branch would not yield any resource savings. In order for our metareasoner to be useful, it must be able to *estimate* the prediction quality for each node without fully calculating prediction quality.

The metareasoner uses an abstraction[3] of the world states such that every world state belongs to one of a comparatively smaller number of the *contexts* in the abstraction. As the agent makes decisions within the environment, the behavioral predictions made by a model and the resource usage (CPU load and total time, for example) are monitored by the metareasoner, and accumulated in the abstract context associated with the prediction's real world state. The metareasoner also monitors the target agent's observed behavior and accumulates the probability distribution in the abstract context representing the world state the observation was made in. Whenever the agent needs to determine the value of a proposed action, the metareasoner first updates the relative prediction utility of each model in each abstract context using the collected data from past predictions and observations. The matrix of estimated prediction utilities (utility for each model in each context) is then passed to the object-level action value calculator. The object-level can then choose the highest-utility models for each prediction node visited during the recursive value calculation over sub-branches from the current world state in the game tree.

[3]While the method of generating the abstraction is beyond the scope of this work, there is much existing work on automated abstraction techniques. For example, see (Gilpin, Sandholm, & Sorensen 2007)

We now define the architecture of a metareasoning agent designed to act within the multi-agent environment described previously. At the highest level, the system is one in which the meta-level device is monitoring the performance of the object-level reasoner and computing the utility of reasoning methods, as depicted in Figure 1.

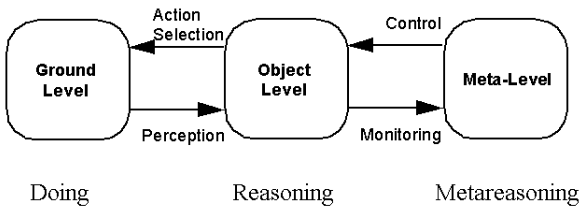

Figure 1: Overview of a Metareasoning-controlled agent. This diagram originally used in (Cox & Raja 2007)

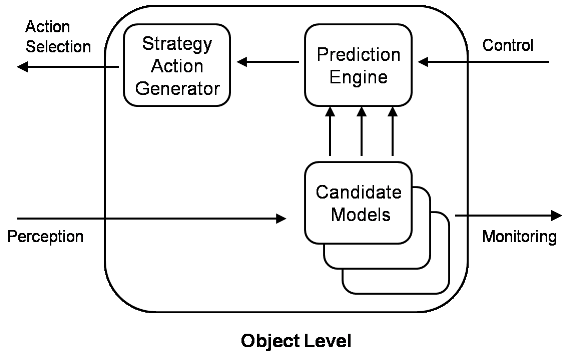

Figure 2: Object Level. The object's candidate models observe the action of other agents and provide predictions to a prediction engine. The prediction engine is controlled by metareasoning, based on the meta-level monitoring of the performance of the candidate models.

We look inside the reasoning object in Figure 2. There are a number of candidate models which receive input from the environment (and observe the actions of the other agent[4]). We assume that each candidate model is a black-box: it is opaque to us and it takes inputs and produces outputs in the form of a probability distribution over the possible actions the modeled agent could take. Furthermore, we assume that the models are input-heterogenous but output-homogenous: each model is trying to predict a distribution of what the modeled agent will do next, but each may be considering different information from which to make their predictions.

At the object level, the Strategy Action Generator is attempting to discover fruitful strategies for future behavior of the agent. In order for the Strategy Action Generator to

[4]For the sake of linguistic clarity, we will refer to only one agent being modeled to avoid confusion between the agent and the multiple candidate models used to predict the actions of that agent. While not discussed further in this work, the metareasoning arrangement proposed here can be used to model multiple target agents.

choose a high-expected-value future strategy (sequence of actions and responses by other agents) it may need to consider many "what if" predictions about various events that could occur in the future in order to evaluate different possible strategies. These predictions vary, depending on the abstract context (collection of world states) they occur in. Depending on the context of the desired prediction, some models may have better prediction performance than others. Thus, the selection of a predictive model (or weighted distribution over models) should be *context-aware* for *each* prediction. For each prediction-in-context that must be made, it is the Prediction Engine's job to generate the overall prediction using the set of available candidate models.

The Prediction Engine's selection of model(s) is influenced by the metareasoning level. The metalevel shown in Figure 3 is composed of several modules.

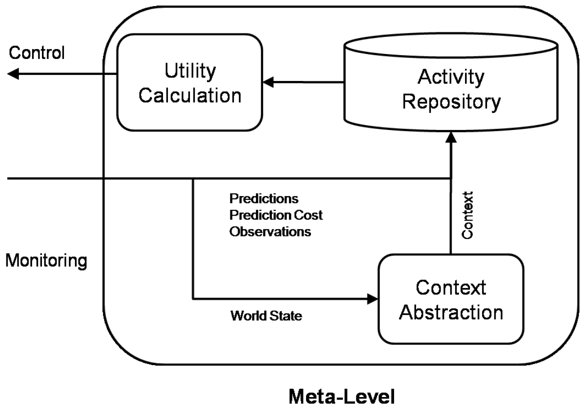

Figure 3: Meta-Level. The metareasoner reviews past predictions, prediction costs, and observations to yield a utility measurement for each context. The utility calculation advises the prediction engine by providing a utility profile for each model in each context.

The first module, the Context Abstraction is a device which determines the appropriate abstract context from a given world state. It provides this context as a tag for the Activity Repository for every prediction and observation event. Whenever a model is queried for a prediction in the object level, it generates a prediction event. Prediction events contain the predicted distribution over modeled agent actions and the cost incurred by that model to generate the prediction. Observation events include the observed behavior of the modeled agent. The Utility Calculation first computes the distribution of agent behavior in each tagged context. Then it calculates a prediction quality measurement from the predictions and observations seen within the context. Finally it computes the utility for each model in each context from the prediction quality and the prediction cost. The latest utility profile matrix is then passed to the Prediction Engine prior to each prediction.

The remainder of this paper presents and explores one of the important facets of this metareasoning architecture: measuring the prediction performance of a model in a context. Before we discuss our novel and theoretically-grounded measurement concept, we review the related research in the area.

Related Work

Agent modeling is often used in computational analysis of an environment to help an agent make decisions. Researchers have examined many domains where agent modeling can be useful: competitive economics, national security, politics, and, of course games (Kott & McEneaney 2007).

Computer science is rich with research on different techniques for how to represent an agent in a model, how to obtain the information to populate the model, how to use the model, and how to evaluate the model based on overall performance in the domain. While actual performance within the domain is ultimately the goal for agent, there are many other factors affecting overall agent performance besides how accurate the agent model is. The structure of the environment, the prior biases in the behavior of the modeling agent, and the way it uses the information from the model are a few examples. If we want to determine *how well a model works* in order to decide which of several models is better, we need to measure the accuracy of the models *directly*.

While most researchers have provided empirical studies that compare the overall domain performance of their methods with other modeling methods or different approaches (such as Monte-Carlo simulation, game-theoretic equilibrium play, or rules-based strategy), few have quantified how well their models predict the other agents' behavior directly.

There are several exceptions in which researchers do examine the accuracy of the model, not just the performance of the overall system. Carmel and Markovitch examine the model size and average error versus sample size while running their domain independent modeling US-*L** algorithm (Carmel & Markovitch 1996b), showing that model size growth slows with more examples while average error drops. Rogowski expands on this work, providing an algorithm *it-us-l** and presents its domain-independent model quality measure in several experiments: average hold-out-set prediction accuracy (Rogowski 2004). In robot soccer games in RoboCup, Riley and Veloso use the probability of correctly recognizing which play an agent is about to make to measure their learning algorithm (Riley & Veloso 2002) but do not provide any other direct model quality measures. In the plan recognition field researchers have also employed the measurements of precision and recall (Blaylock & Allen 2005; Cox & Kerkez 2006) when comparing performance of candidate recognition algorithms.

The majority of the model prediction quality measures in these efforts rely on extensions to error measurements intended for binary classification (1 point for correct prediction, 0 points for incorrect prediction). While this information does provide a basic quality measure, its value diminishes as the number of possible agent actions per state grows. In the world described in the previous section, there are no restrictions on uniformity of the number of actions leading from a node. Some nodes may have few actions leading away while others may have many. Under this condition, the results of a function comprised of binary-based prediction quality measurements from multiple heterogenous nodes is unclear. For the function to be meaningful, the underlying quality measure must be more universal.

A more detailed representation of general classification performance (which is applicable to the performance measurement of a model that predicts which action an opponent will take) is the confusion matrix. A confusion matrix is an M-by-M matrix which represents a classifier's distribution of classification labels provided to M different classes. Column headings hold the model's predictions and row headings indicate the true class. Thus, the cells along the diagonal represent correct classifications and the cells on the off-diagonal represent incorrect classifications. In addition to providing accuracy or error rate (computed from the main diagonal of the matrix), the matrix reveals the number (or probability) of each type of mistake (off-diagonal cells). This additional information can be valuable when different predictive mistakes have different costs. Several agent modeling efforts use the confusion matrix to characterize their model prediction quality. Davidson uses a confusion matrix to quantify a neural-network agent model's ability to predict whether the opponent will fold, call, or raise under many different circumstances in poker (Davidson *et al.* 2000). Sukthanar and Sycara use a confusion matrix to characterize their prediction of which type of breach and enter maneuver the opposing force is about to perform in a simulated military tactical engagement (Sukthankar & Sycara 2006).

While confusion matrices are a step in the right direction in that they provide quality assessments beyond basic binary prediction accuracy, they can quickly become unwieldy if the space of possible options (M) is large (or continuous) or there are multiple stages of predictions that must be made during the course of an engagement (such as in chess, poker, and military endeavors). In multiple-stage prediction encounters (which can often be characterized as extensive form games and depicted with game trees) there would need to be a confusion matrix for every possible agent prediction node. Sometimes the act of getting to a prediction node is not certain - the probability depends on which events occurred over the history of the encounter (the path taken through the game tree). The probabilities might be co-dependent (depending on the probabilities of all the agents in the environment). In these circumstances there may be no clear way in which to generate a single confusion matrix quantifying the goodness of a model.

A New Approach For Evaluating Prediction Performance In Agent Models

Given the difficulty of the problem and the lack of its treatment in the field, we now present a flexible method for characterizing relative prediction quality that is independent of the agent in which it resides. This quality measurement does not consider total performance of the agent - only the model's ability to predict another agent's behavior. Before we present the measurement, let us first take a moment to list the desiderata of a predictive performance measurement that is to be used in a metareasoning role.

Given a *context* (an abstraction of the world state), an actual behavior distribution (a probability distribution over a finite number of possible actions[5], which describes the behavioral choices of the agent in that context) and a prediction of the target's behavior distribution in the context (as provided by a predictive model, given the context), we define *prediction divergence* ($\mathcal{PD}(P,Q)$) as a scalar measure of the distance between the actual distribution P and the predicted distribution Q such that $\mathcal{PD}(P,Q)$ has the following properties:

1. $\forall P,Q \;\; \mathcal{PD}(P,Q) = 0 \Leftrightarrow P \equiv Q$. The function value should be zero if and only if the predicted behavior distribution is the same as the actual behavior distribution. This is a necessary condition for the function to be used to measure error or loss.

2. $\forall P,Q \;\; \mathcal{PD}(P,Q) > 0 \Leftrightarrow P \neq Q$. The function value should be greater than zero if and only if the behavior distribution is different than the predicted distribution. This is a necessary condition for the function to be used to measure error or loss.

3. $\forall P,Q \;\; \mathcal{PD}(P,Q) = \mathcal{PD}(Q,P)$. The function is symmetric. There is no sensitivity to which distribution represents the truth and which is the prediction. This is a necessary condition for the function to be a metric.

4. $\forall P,Q,R \;\; \mathcal{PD}(P,Q) + \mathcal{PD}(Q,R) \leq \mathcal{PD}(P,R)$. The function obeys the triangle inequality, enforcing the logical understanding of "distance" in metric space. When combined with the previous three properties, this property completes the characterization of the function as a metric. While not essential for the work presented in this paper, we require this property to facilitate future work in the area.

5. $\forall P,Q,R,S \;\; (\mathcal{PD}(P,S) > \mathcal{PD}(P,R)) \wedge (\mathcal{PD}(P,R) > \mathcal{PD}(P,Q)) \Rightarrow \mathcal{PD}(P,S) > \mathcal{PD}(P,Q)$. The function values obey the transitive property. Distributions which are further apart should have a greater function value than those which are closer together[6].

6. Given any *extreme*[7] distribution P and any of its distributional complements[8] P^{-1}, $\mathcal{PD}(P,P^{-1}) = C$ where C is a positive constant. The function values should be bounded by a fixed constant such that any two distribu-

[5]While we describe a finite number of actions here, the functions presented later are applicable to continuous distributions as well.

[6]The notion of "further apart" in distribution terms can be expressed with an example. Consider a fair coin with distribution $P(heads) = 0.5$ and $P(tails) = 0.5$. This coin's distribution is closer to a loaded coin with distribution $P(heads) = 0.4$ and $P(tails) = 0.6$ than it is to one with distribution $P(heads) = 0.1$ and $P(tails) = 0.9$.

[7]An extreme distribution is on in which one of the elements contains all of the probability mass and the other elements all contain no probability mass.

[8]We define a distributional complement for an extreme distribution as another extreme distribution where all of the probability mass is located at a different point than in the original distribution.

tions which are maximilally far apart[9] are bounded. This enables the function to be scaled by dividing by C such that the function values lie on the interval $[0,1]$.

7. $\forall$ extreme P, Q, and their respective complements P^{-1}, Q^{-1} $\mathcal{PD}(P, P^{-1}) = \mathcal{PD}(Q, Q^{-1})$. All function bounds are equal regardless of the width of the underlying distributions. This is a minor extension of the previous property which allows heterogenous nodes to be weighted and summed in mathematically meaningful ways. If the function yielded differing values for distributions of different widths, it would be impossible to compare prediction qualities at two decision nodes where the number of actions to choose from differed.

Comparing Behavior Models with Real Agent Behavior We now develop the measurement function characterized by the desired properties described above. Information theory provides tools for comparing distributions. Its tools are backed by proven theory and have been in use for well over half a century. Relative entropy (also known as the Kullback-Leibler Distance) between two probability mass functions (Cover & Thomas 2006) is a very widely used information theoretic similarity measure. Given two distributions p and q, *relative entropy* is defined as:

$$D(p\|q) = \sum_{x \in \mathcal{X}} p(x) \log \frac{p(x)}{q(x)} \tag{1}$$

While relative entropy is characterized by certain desirable properties such as equality, non-negativity, and transitivity, unfortunately it is not symmetric (Property 3) or bounded (Property 6). We instead choose a related, but lesser known similarity measure which has the additional properties of being both symmetric and bounded as discussed in (Fuglede & Topsoe 2004; Topsoe 2000). This measure is known as Jensen-Shannon Divergence (*JSD*) and is sometimes referred to as Capacitory Discrimination. *JSD* is calculated between two distributions p and q as:

$$JSD(p,q) = D\left(p \middle\| \frac{(p+q)}{2}\right) + D\left(q \middle\| \frac{(p+q)}{2}\right) \tag{2}$$

One can think of *JSD* as the average relative entropy from each distribution (p and q) to the distribution that is midway between them. The further the distributions are apart, the higher their *JSD* value. In addition to keeping the desired properties from relative entropy, the square root of *JSD* is a metric (Topsoe 2000), meaning that it covers the first four properties above and provides additional validity for its use as a distance measurement. For the remainder of this work we will use the term *Root-JSD* to represent the square root of the value calculated in Equation 2. By letting one of the distributions represent the predicted behavior distribution produced by the model and letting the other distribution represent the observed behavior distribution of the target agent, we can measure the predictive quality of our model using *Root-JSD*.

[9] A loaded coin with $P(heads) = 0$, $P(tails) = 1$ is maximally far from a loaded coin with $P(heads) = 1, P(tails) = 0$.

For any given interaction between agents, we may have many contexts. Each context consists of many pairs of predicted and observed actual behavior distributions that need to be compared using *Root-JSD*. For example, in extensive form games such as poker, a model might be based on a portion of the game tree which contains a set of the opponent's decision nodes (N). At each node ($n \in N$) where the opponent has an opportunity to decide his next action there exists a distribution over his possible actions. Some nodes may be occur more frequently or may be worth more (in terms of risked utility). When developing a prediction quality measurement we would like to entertain the concept of weighting the prediction quality at each decision node by its importance in our decision-making and then combining the weighted prediction quality for all the nodes of interest.

In order to measure the overall similarity between the set of predicted conditional probability distributions (P) and the set of observed conditional probability distributions (Q) in an interaction, we define a function that combines all of the individual *Root-JSD* measurements at each agent-decision node in the set, using a weighting function for each node. Given a set of opponent decision nodes (N), associated sets of predicted and true probability distributions $\forall k \in N,\ P_k,\ Q_k$, and a weighting function for each node $\forall k \in N,\ W(k)$, we define *Weighted Prediction Divergence (WPD)* as:

$$WPD(N) = \frac{\sum_{k \in N} W(k) \sqrt{JSD(P_k, Q_k)}}{\sum_{k \in N} W(k)} \tag{3}$$

Equation 3 yields a value in $[0,1]$ making it very useful for comparing overall prediction quality of multiple models for entire histories of agent interactions. Using this function, we can also explore subsets of decisions that fit into the definition of a particular context group as described previously. We can weight the individual decision nodes within the context group according to some function of their *importance* with respect to the decision problem. Some possible weighting functions and their rationale for use include:

- Uniform Weighting. When there is no justification for weighting prediction nodes differently they are all given a weight of one.
- Frequency Weighting. When decision nodes have different frequencies of appearing in an interaction, it may make sense to weight the nodes by their frequency of occurrence. This makes the quality assessment sensitive to decisions that have to be made more often.
- Utility Weighting. Some decisions are more valuable than others: the outcome of a valuable decision leads to more increase or decrease in utility for the modeling agent. In these cases it may be desirable to weight the nodes by a function of the possible utility outcomes (for example, $U_{\max} - U_{\min}$).
- Risk (Reward) Weighting. Multiplying the probability of a node occurring with the utility weight it is characterized by yields its risk. In this weighting scheme, the higher the risk, the higher the weight.

In the next section, we use a modified frequency weighting scheme to compare the performance of two black-box predictive models (fixed and learning) in the Texas Hold'em poker domain.

Experiments

We now show how the predictive performance measure can be used to differentiate model performance in different contexts. While these experiments were conducted outside of a metareasoning agent, they demonstrate the validity of the technique described previously for determining which model is better at making predictions within each of several contexts. In the remainder of this section we introduce our target domain and describe the models which we are comparing and the protocol for the measurement. We then show the results of the experiments and discuss the implications for the feasibility of metareasoning using an automated version of this technique.

Texas Hold'em Domain Characteristics

Texas Hold'em poker is a well known zero-sum imperfect information game. It is a poker variant in which players attempt to make the best set of cards from their private (hole) cards and several fully visible community cards which are usable by every player. Each hand consists of multiple rounds of dealing and betting where players try to win the pot: the hand ends when either all but one player has folded (uncontested win) or several players have stayed in the game until all betting is complete (the showdown). In an uncontested win, the player remaining after all others fold wins the pot. In the showdown, the player with the highest valued set of cards wins the pot.

There are four rounds in which players can bet chips based on the strength of their cards: the *pre-flop*, *flop*, *turn*, and *river*. If any player decides to fold before all four betting rounds have been completed, he forfeits any claim on the pot and must sit out the remainder of the hand while the other players finish. If all but one of the players have folded prior to the completion of the river betting round then the remaining player wins the pot uncontested and the hand is terminated without further betting rounds.

Each round has a different set of rules regarding the way bets are made. On the pre-flop, players bet knowing only the two private cards they hold since no community cards have been dealt yet. After the pre-flop betting is over, the dealer places three community cards on the table and players make another round of betting during the flop based on the overall strength of their private cards and the shared community cards. After the flop betting is complete, the dealer places one more community card on the table and another round of betting occurs during the turn. Next, the dealer places a final community card on the table and the players bet during the river. If more than one player remains after the river betting round is complete then the showdown occurs.

Texas Hold'em uses *blinds* to offset the information advantage of players who act later in the pre-flop. Blinds are compulsory payments from some players to the pot before any cards are dealt. In general, the player who will act last in the pre-flop pays a fixed amount of money into the pot and the player who will act just before him pays $1/2$ that amount. These amounts are known as the big blind and little blind respectively.

In a heads-up (two player) limit (fixed increment bet) Texas Hold'em match, we can express each round of betting in terms of an extensive form game. In this version of Texas Hold'em there are over 10^{17} (Billings *et al.* 2003) possible game outcomes - making even this simplistic variant of Texas Hold'em very rich in strategic opportunity.

Opponent Modeling in Poker

There are two types of opponent models for the poker domain: fixed and learning. A fixed model is usually based on either general poker knowledge or an offline-trained program that reviews previously documented matches and builds a model from them. The fixed model doesn't change its function over time during an encounter with an adversary. A learning model is different from a fixed model in that it can change its strategy over time as it observes the behavior of its current adversary during an encounter.

For these experiments we compare the performance of two black-box models: a fixed strategy opponent model and a learning opponent model. Each of these models were part of a larger modeling system in our entry for the 2007 Computer Poker Competition: PokeMinnLimit1. The fixed strategy model was based on general poker knowledge for playing limit poker. It uses static parameters for a pot-value function that predicts the likelihood of the opponent folding, calling or raising. In competition, our agent uses the fixed model to provide predictions of opponent behavior before there are sufficient observations to use the learning model.

In contrast, the learning model has no pre-defined probability distribution over opponent behavior. Instead, it attempts to make predictions of the opponent's behavior based solely on past observations in the current match once at least one observation has been made at that decision node. At each node in the game tree where the opponent makes a decision the learning model records the frequencies of the actions. Then it computes the probability that the opponent will fold, call or raise in the future based on the accumulated frequencies of each action, conditional on the game tree node where the agent made the decision. This particular model makes several assumptions in order to reduce the number of nodes in the game tree which it must compute: the model does not consider what cards the opponent may have; each betting round is considered independent - thus the model is actually carrying four separate game trees of information - one for each round. There is no windowing or time-based decay in this model - all data is gathered and weighted equally for the entire course of the encounter. This particular style of model is biased towards learning a stationary policy opponent well, at the possible expense of being mislead by a non-stationary adversary.

Comparing Model Quality

We now show how *WPD* can be used to compare performance between two candidate agent models. We focus on the models' ability to predict the behavior of the top performer in the 2007 Computer Poker "Limit Equilibrium" Competition: Hyperborean07LimitEq1.

The modified frequency weighting function $W(n)$ we use in these experiments to compute *WPD* weights each node by how *relevant* it is in the poker game tree. We calculate this relevance recursively: the root-node of a tree has a relevance of 1. Each child of the root-node has a relevance of $\frac{1}{|C_r|}$ where $|C_r|$ is the number of the root-node's children. Each child of the root-node's children C_c is has a relevance of $\frac{1}{|C_r||C_c|}$ and so forth.

We compared the performance of the fixed model to the learning model. To compare the prediction quality of these models we needed data about the true behavior of our desired opponent. For the test-data set, we retrieved all 660 matches (1.98M hands) from the 2007 Computer Poker Competition (Zinkevich 2007) played by Hyperborean07LimitEq1. We felt this data was representative of the true behavior of Hyperborean07LimitEq1 because it was the largest set of data available to the public and the data documents Hyperborean07LimitEq1's behavior against many different opponents. We scanned all of the test data to determine the frequency counts for each action at each game tree node in each round. We declared the resulting probability distributions for each node in the game tree as behavioral truth for Hyperborean07LimitEq1. To determine the learning model's ability to learn the opponent's distribution of behavior, we isolated a training set of 20 matches played between PokeMinnLimit1 and Hyperborean07LimitEq1. We trained the learning model using only observations that it could have made during these 20 matches. We then gathered predictions from the trained model on the hands from the test set and calculated the *WPD* between the learning model's predictions of Hyperborean07LimitEq1's behavior and Hyperborean07LimitEq1's actual behavior. A similar procedure (without a training phase) was used to obtain the *WPD* for the fixed-strategy model. The aggregate results for 3000 hands for both models are shown in Figure 4. Because of the heterogeneous nature of Texas Hold'em poker's four betting rounds, we chose to separate the opponent decisions into four context groups: pre-flop, flop, turn, and river. While we made this decision based on our expertise of the domain, we expect that there may be better divisions of contexts that could be made (by pot size, for example) which might yield even further separation between the different models. The automation of context determination is a key area for future research in order to make further progress with the metareasoning system described in this paper.

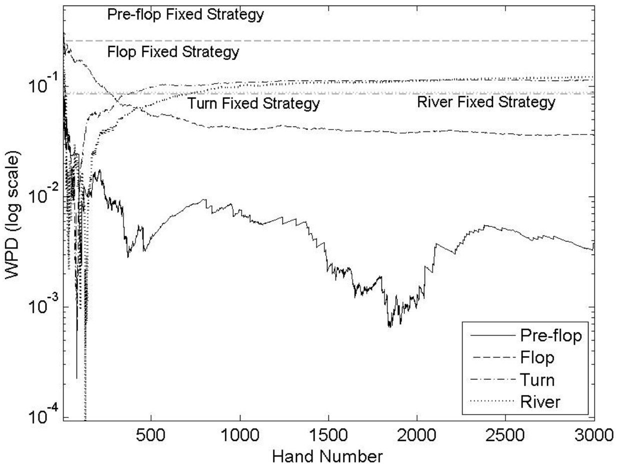

Figure 4: Comparing Learning versus Fixed Strategy models in each of the four betting rounds in a 3000-hand limit poker game. The horizontal lines represent the fixed-strategy model *WPD*. The darker fluctuating lines represent the learning model's *WPD* values.

Discussion

While the learning model quickly becomes better at predicting behavior than the fixed-strategy model during the preflop and flop, the learning model's prediction quality in the turn and river grow worse over the course of the game to the point where the fixed-strategy model would have better predictive performance after approximately 380 iterations on the turn (and after approximately 740 iterations on the river).

While the learning model appears to be very useful for predicting the true opponent's behavior on the pre-flop and possibly the flop (at least when compared to the fixed model we used), its utility is questionable during the turn and river, where the fixed model's predictions might be preferred. Although finding the cause of the reduced accuracy of the learning model during the turn and the river is beyond the scope of this work, it is important to note that this analysis reveals the flaws in the learning model even when other performance based methods fail to do so. For example, in a separate experiment with 100 matches (300,000 hands) we examined relative performance between two agents in terms of number of small bet increments won per hand (sb/h). We noticed that the learning strategy model achieves an average 0.04 sb/h greater win rate than the fixed strategy model from hands 500 through 3000 when each played against Hyperborean07LimitEq1. If we had relied on just the earnings performance measure as the method of determining the better model (as much of the other research in the field does), we might not have realized that subcomponents of the learning model were not performing as well as the same subcomponents in the fixed model.

If the meta-level reasoning system described earlier had been available at the time of competition and could have seen the prediction quality difference between the learning and fixed models in these contexts, it might have been possible for the recommender to advise the agent to switch to using the fixed model instead of the learning model in the turn and river betting rounds. This may have improved the performance of the agent in the competition. Testing this hypothesis is reserved for future work.

Conclusion and Future Work

We have shown how prediction quality assessments can be used to point out strengths and weaknesses in opponent models that remain hidden under domain-based performance measurements. We identified the desiderata of a strong measure of model quality necessary for metalevel reasoning in the multi-model agent prediction setting. By borrowing a concept from information theory (Jensen-Shannon Divergence) we've developed a Weighted Prediction Divergence metric and characterized several tailored weighting schemes. Weighted Prediction Divergence incorporates all desired properties of our performance measure allowing us to determine the relative prediction quality between models. We have shown empirically how Weighted Prediction Divergence values for several models could be used to select the best model for the current context. These techniques form one of the foundations of metareasoning, empowering a meta-level reasoner to make real time decisions about which models to use in certain contexts.

There is at least one additional effort that must be completed before the metareasoner described in the introduction can be realized. We must choose a state abstraction method which can automatically cluster world states into contexts such that the prediction quality estimation system described in this work can assess models with fewer computational resources. While we have shown a meaningful context for poker in this work, the contexts for the collection of nodes we used were generated manually. Automating this process would enable multi-domain applicability for the metareasoning technique we present.

Acknowledgements

We would like to thank the entire University of Minnesota Poker Research Group for their help and support with this effort. The views expressed in this article are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the U.S. Government.

References

Billings, D.; Burch, N.; Davidson, A.; Holte, R.; Schaeffer, J.; Schauenberg, T.; and Szafron, D. 2003. Approximating game-theoretic optimal strategies for full-scale poker. In *International Joint Conference on Artificial Intelligence*, 661–668.

Blaylock, N., and Allen, J. 2005. Recognizing instantiated goals using statistical methods. In Kaminka, G., ed., *IJCAI Workshop Modeling on Others from Observation (MOO-2005)*, 79–86.

Carmel, D., and Markovitch, S. 1996a. Incorporating opponent models into adversary search. In *Thirteenth National Conference on Artificial Intelligence*, 120–125.

Carmel, D., and Markovitch, S. 1996b. Learning models of intelligent agents. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, 62–67.

Cover, T. M., and Thomas, J. A. 2006. *Elements of Information Theory*. Wiley-Interscience, 2 edition.

Cox, M. T., and Kerkez, B. 2006. Case-based plan recognition with novel input. *Control and Intelligent Systems* 34(2):96–104.

Cox, M. T., and Raja, A. 2007. Metareasoning: A manifesto. Technical Report TM-2028, BBN Technologies, Cambridge, MA. www.mcox.org/Metareasoning/Manifesto.

Davidson, A.; Billings, D.; Schaeffer, J.; and Szafron, D. 2000. Improved opponent modeling in poker. In *International Conference on Articial Intelligence*, 1467–1473.

Fuglede, B., and Topsoe, F. 2004. Jensen-shannon divergence and hilbert space embedding. In *Proceedings of the International Symposium on Information Theory, 2004. (ISIT 2004)*, Page 31.

Gilpin, A.; Sandholm, T.; and Sorensen, T. B. 2007. Potential-aware automated abstraction of sequential games, and holistic equilibrium analysis of texas hold'em poker. In *Twenty-second National Conference on Artificial Intelligence*, 50–57.

Kott, A., and McEneaney, W. M. 2007. *Adversarial reasoning : computational approaches to reading the opponent's mind*. Boca Raton: Chapman & Hall/CRC.

Luckhardt, C. A., and Irani, K. B. 1986. An algorithmic solution of n-person games. In *AAAI-86: Proceedings of The Fifth National Conference on Artificial Intelligence*, 158–162.

Riley, P., and Veloso, M. 2002. Recognizing probabilistic opponent movement models. In *RoboCup 2001: Robot Soccer World Cup V*. Springer Verlag.

Rogowski, C. 2004. Model-based opponent-modelling in domains beyond the prisoners dilemma. In *Workshop on Modeling Other Agents from Observations at AAMAS*, 41–48.

Russell, S. J., and Norvig, P. 2003. *Artificial intelligence: a modern approach*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., second ed. edition.

Russell, S., and Wefald, E. 1991. Principles of metareasoning. *Artificial Intelligence* 49(1-3):361–395.

Stone, P.; Riley, P.; and Veloso, M. M. 2000. Defining and using ideal teammate and opponent agent models. In *Twelfth Innovative Applications of AI Conference (IAAI-2000)*, 1040–1045.

Sturtevant, N.; Zinkevich, M.; and Bowling, M. 2006. $prob - max^n$: Playing n-player games with opponent models. In *National Conference on Artificial Intelligence (AAAI)*, 1057–1063.

Sukthankar, G., and Sycara, K. 2006. Robust recognition of physical team behaviors using spatio-temporal models. In *Proceedings of Fifth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*.

Topsoe, F. 2000. Some inequalities for information divergence and related measures of discrimination. *IEEE Transactions on Information Theory* 46(4):1602–1609.

Zinkevich, M. 2007. Data files from computer poker competition. Internet (Unreviewed). (http://www.cs.ualberta.ca/˜pokert/2007/data.html).