Bayes-Relational Learning of Opponent Models from Incomplete Information in No-Limit Poker

Marc Ponsen\textsuperscript{1}, Jan Ramon\textsuperscript{2}, Tom Croonenborghs\textsuperscript{3}, Kurt Driessens\textsuperscript{2} and Karl Tuyls\textsuperscript{4}
\textsuperscript{1}: MICC Maastricht University, Netherlands
\textsuperscript{2}: Declarative Languages and Artificial Intelligence Group, Katholieke Universiteit Leuven (KUL), Belgium
\textsuperscript{3}: Biosciences and Technology Department, KH Kempen University College, Belgium
\textsuperscript{4}: Technische Universiteit Eindhoven, Netherlands

Abstract

We propose an opponent modeling approach for no-limit Texas hold-em poker that starts from a (learned) prior, i.e., general expectations about opponent behavior and learns a relational regression tree-function that adapts these priors to specific opponents. An important asset is that this approach can learn from incomplete information (i.e., without knowing all players’ hands in training games).

2. Learning an Opponent Model

We learn an opponent model for players in the game of No-Limit Texas Hold’em poker. To make the model useful to an AI player, we must be able to learn this model from a limited amount of experience and (if possible) adapt the model quickly to changes in the opponent’s strategy. An added, and important, difficulty in poker is that we must be able to learn this model given a large amount of hidden information. We propose to start the opponent model with a prior distribution over possible action choices and outcomes. We will allow the model to adapt to different opponents by correcting that prior according to observed experience.

Consider a player \( p \) performing the \( i \)-th action \( a_i \) in a game. The player will take into account his hand cards, the board \( B_i \) at time point \( i \) and the game history \( H_i \) at time point \( i \). The board history specifies both the identity of each card on the table (i.e., the community cards that apply to all players) and when it appeared, and \( H_i \) is the betting history of the game for all players. The player can fold, call, or bet.

We limit the possible outcomes of a game \( r_p \) for a player \( p \) to: 1) \( p \) folds before the end of the game (\( r_p = \text{lose} \), 2) \( p \) wins without showing his cards (\( r_p = \text{win} \)) and 3) \( p \) shows his cards (\( r_p = \text{cards}(X,Y) \)). This set of outcome values also allows us to learn from examples where we did not see the opponent’s cards, registering these cases as \( \text{win} \) or \( \text{lose} \), without requiring the identities of the cards held by the player. The learning task now is to predict the outcome for an opponent \( P(r_p|B_i, H_i) \) and the opponent action (given a guess at his hand cards) \( P(a_i|B_i, H_{i-1}, r_p) \).

2.1. Learning the Corrective Function

We propose a two-step learning approach. First, we learn general functions predicting outcomes and actions for poker players in general. There is a huge amount of freely available training data for such functions. These functions are then used as a prior, and we learn a corrective function to model the behavior and statistics of a particular player. The key motivations for this are first that learning the difference between two distributions is an elegant way to learn a multi-class classifier (e.g. predicting distributions over \( 2^4 \cdot (52^2/2) \) possible outcomes) by generalizing over many one-against-all learning tasks, and second that even

\textsuperscript{1}A longer version of this paper will appear in the Proceedings of AAAI-2008.
with only a few training examples from a particular player already rather accurate predictions are possible.

In the sequel, we mean with an example a tuple \((i, p, a_i, r_p, H_{i-1}, B_i)\) of the action \(a_i\) performed at step \(i\) by a player \(p\), together with the outcome \(r_p\) of the game, the board \(B_i\) and the betting history \(H_{i-1}\).

Consider the mixture \(D_{p+i}\) of two distributions: the distribution \(D_*\) of arbitrarily drawn examples from all players and the distribution \(D_p\) of arbitrarily drawn examples from a particular player \(p\). Moreover, consider the learning problem of, given a randomly drawn example \(x\) from \(D_{p+i}\), predicting whether \(x\) originated from \(D_*\) or from \(D_p\). For a given learning setting (either predicting actions from outcomes or predicting outcomes from actions), it is easy to generate examples from \(D_*\) and \(D_p\), labeling them with \(*\) or \(p\), and learning the function \(P(D_p|x)\), giving for each example \(x\) the probability the example is labeled with \(p\). We do so using the relational probability tree learner TILDE (Blockeel & De Raedt, 1998). From this learned ‘differentiating’ model, we can compute the probability \(P(x|D_p)\), for every example \(x\) by using Bayes’ rule:

\[
P(x|D_p) = P(D_p|x) \cdot P(x)/P(D_*)
\]

Since we have chosen to generate as many examples for \(D_*\) as for \(D_p\) in the mixture,

\[
P(D_p) = P(D_*) = 1/2
\]

\[
P(x) = P(D_*|x)P(x|D_*) + P(D_p|x)P(x|D_p)
\]

and substituting (2) and (3) into (1) gives:

\[
P(x|D_p) = \left( P(D_p|x) \cdot \left( \frac{1}{2}P(x|D_p) + \frac{1}{2}P(x|D_*) \right) \right) \cdot \frac{1}{2}
\]

\[
= \frac{P(D_p|x)p(x|D_p)}{1 - P(D_p|x)}
\]

From this, we can easily get:

\[
P(x|D_p) = \frac{P(x|D_*) \cdot P(D_p|x)}{1 - P(D_p|x)}
\]

Here, \(P(x|D_*)\) is the learned prior and \(P(D_p|x)\) is the learned differentiating function.

Having now explained how to learn a player-specific prediction function given a prior, the question remains as how to learn the prior. We learn the prior by (again) learning a differentiating function between a uniform distribution and the distribution formed by all examples collected from all players. Even though the uniform distribution is not accurate, this is not really a problem as sufficient training examples are available.

3. Experiments and Results

We observed cash games (max 9 players per game) played in an online poker room and extracted examples for players who played more than 2300 games. We randomly selected 20% of the games for the test set. To evaluate our learning strategy, we report the log-likelihoods of the learned distributions and compare this with reasonable priors. A model with a higher likelihood directly allows an algorithm to sample and estimate the outcome of a game more accurately.

3.1. Estimation of Action and Outcome Probabilities

We learned both priors and player-specific functions for each of the four phases (preflop, flop, turn, river) of the game. Figure 1 plots loglikelihoods averaged over 8 players for different training set sizes. The priors are clearly better than uninformed priors (i.e. not using learning). After having observed 200 games, in general the likelihood improves with the size of the training set.

![Figure 1. Experiment predicting the action, given outcome, board and game history.](image)

4. Conclusions and Future Work

We presented a Bayes-relational opponent modeling system that predicts both actions and outcomes for human players in the game of No-Limit Texas Hold’em poker. Both these sources of opponent information are crucial for simulation and game-tree search algorithms, such as the adaptive tree search method by (Billings et al., 2006). The Bayes-relational opponent modeling approach starts from prior expectations about opponent behavior and learns a relational regression tree-function that adapts these priors to specific opponents. Our experiments show that our model adapts to specific player strategies relatively quickly.

References

