Using Bayesian Networks to Model a Poker Player

Andrew Heiberg
University of California, San Diego
aheiberg@cs.ucsd.edu

Abstract

Opponents are characterized by a Bayesian network intended to guide Monte-Carlo Tree Search through the game tree of No-Limit Texas Hold'em Poker. By using a probabilistic model of opponents, the network is able to integrate all available sources of information, including the infrequent revelations of hidden beliefs. These revelations are biased, and as such are difficult to incorporate into action prediction. The proposed network mitigates this bias via the expectation maximization algorithm and a probabilistic characterization of the hidden variables that generate observations.

1 Introduction

The Monte-Carlo approach to navigating a stochastic, partially observed environment such as poker is to repeatedly simulate random gameplay until confident estimates of the rewards can be established. Better estimates can be achieved in fewer samples if the stochastic nature of the environment being explored is accurately characterized. For poker, this means treating opponents as part of the environment. An accurate opponent model can direct the search to more relevant parts of the tree sooner and more frequently, yielding better estimates of the expected values for different potential actions.

In the context of Monte-Carlo Tree Search (MCTS), an opponent model consists of two distributions. The first specifies how likely the opponent is to take a particular action given the current state. The task is to learn $\mathbb{P}(a_n|a_1...a_{n-1}, \mathbf{B})$, where a_i is the i^{th} action taken and $\mathbf{B}$ represents the public cards available to all players.

The second distribution is an estimate of the rewards. When MCTS reaches the edge of the search tree, it simulates the rest of the hand until a terminal game state is reached, at which point the rewards are back-propagated through the visited nodes. If this simulated, terminal state has only one unfolded player, calculating the rewards is straightforword. If this is not the case, a showdown is required to determine the winner, where a showdown consists of each player revealing their hidden cards (hole cards) to determine the winner. The problem, howewer, is that the true hole cards c are not known. In order for rewards to be estimated, the distribution $\mathbb{P}(c|a_1...a_n, \mathbf{B})$ is needed.

1.1 Previous Work

In order to make our contribution clear, a review of how MCTS has previously been applied to poker is needed.

Van den Broeck, Driessens, and Ramon (2009) was the first to apply MCTS to the imperfect information game of poker. The two aforementioned distributions were learned using decision trees. Each action in the dataset (logs of human play in an online casino) was used to train the first distribution, and each showdown was used to train the second. Ponsen, Gerritsen, and Chaslot (2010) followed a similar paradigm, but sophisticated the decision trees and was able to adapt general expectations about opponent behavior to specific opponents.

Kleij (2010) appears to be the first attempt to integrate hole card information into the action distribution $\mathbb{P}(a_n|a_1...a_{n-1}, P, c)$. A predictor of hole cards can be obtained from this distribution via Bayes rule. Learning this distribution, however, presents a problem, because in the small fraction of hands ($\approx 10\%$) that are labeled (i.e. went to showdown), the hole cards are necessarily biased. Consider: players who remain active for all four betting rounds will generally fall into one of two cases: 1) they believe their cards to be winning and call all bets, or 2) they have no faith in their hands and are fortunate enough to check all the way to showdown. If learning were to take place solely on labeled data, the above distribution would never predict a fold. When reversed via Bayes rule, it would predict stronger or weaker hole cards more often than middling hands, which in reality should, by definition, constitute the majority. To mitigate this bias, Kleij (2010) labeled all unlabeled hands with hole cards using a two-stage algorithm inspired by expectation maximization (EM).

1.2 Summary

The opponent model in this paper consists of two Bayesian networks. One is trained to predict beliefs given actions, and the other to predict actions given beliefs.

When predicting beliefs, the only examples from which to learn the network parameters are the hands labeled with hole cards. To mitigate the previously discussed bias associated with these examples (Section 1.1), the network is trained on both labeled and unlabed data using expectation maximization. Predictions about beliefs can then be made by computing the set of beliefs that are most likely given the observed actions. This process also uses pre-computed probabilities of transitioning from one belief to another, further mitigating bias by tethering the computation to the game's statistical ground-truth.

As actions accumulate, the belief network refines the distribution for the current belief of the opponent. Samples from this distribution can be entered as evidence into the action network (also trained via EM) to generate action predictions.

2 Description

The Bayesian network for predicting buckets is depicted in Figure 1. In total, three networks will be used, one for each round of betting (the first round, or preflop, is excluded, as there is not enough information from which to mearningfully learn). The network for the final betting round is shown.

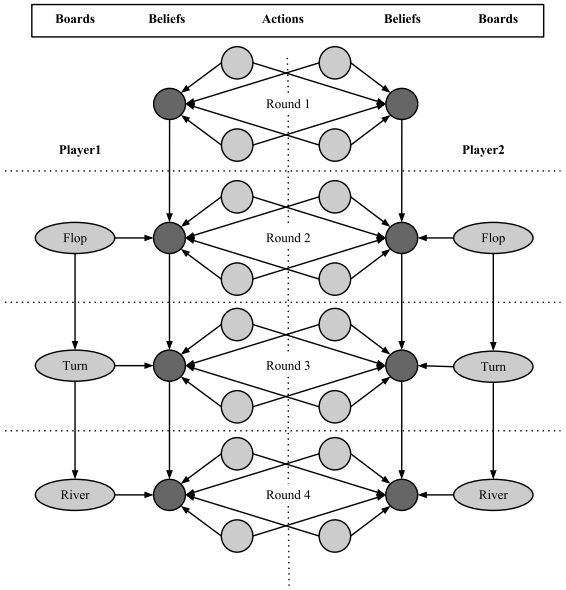

Figure 1: The network for belief prediction, representing a game of two-player, Texas Hold'em poker that has reached the final betting round (also known as the river). Circles and ellipses are nodes, and different node types are organized under the appropriate headings. Belief nodes are darkened to illustrate that they are unobserved.

The network predicting actions is identical for every round, and can be seen in Figure 2.

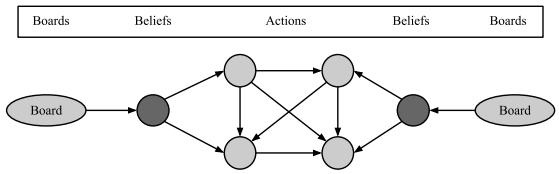

Figure 2: The network for predicting actions from a given bucket and previous actions.

The relationships between three types of nodes must be specified: action, belief and board nodes.

2.1 Action Nodes

The action nodes capture the betting dynamics for each round. Action nodes can take on $n_r + 3 + 1$ values, where n_r is the number of discretized bet:pot ratios allowed (in the No-Limit version of poker, players can bet any amount they wish. In practice, however, bets typically fall close to a few particular fractions of the pot). The three additional actions are 'check', 'call', and 'fold'. Finally, one additional dummy value is included for the betting sequences that terminate before all four action can be taken (e.g both players 'check'). Training hands that exceed this four action limit are thrown out during pre-processing, a weakness considering the network will always predict 'call' or 'fold' for the second action of the second player. The current cap of four is a tolerable simplification given that $\approx 99.97\%$ of training hands do not exceed it.

2.2 Board Nodes

The board nodes represent the public cards as they are revealed throughout the game. Not every board needs to be considered individually: since no suit is more important than any other, many boards are suit-symmetrical to one another. For example, [2h,3h,5d,6d,Tc] is functionally equivalent to [2c,3c,5h,6h,Td]. For round-4, suit-symmetry reduces the number of unique boards from $\mathbb{C}(52,5) \approx 2,600,000$ to $\approx 40,000$, saving both computation and storage.

2.3 Belief Nodes

The belief nodes quantify the unobserved, internal states of the agents that generate the observed action sequences. This quantification is achieved by calculating the expected hand-strength squared value, $E[S^2]$, of that player's hole cards against the given board. For a discussion of $E[S^2]$, see (Johanson, Zinkevich, and Bowling 2007). This range is further discretized by partitioning these values into 10,20,15, and 10 buckets for the round-1, round-2, round-3, and round-4 belief nodes, respectively. Percentile bucketing places the weakest x_1 percent of hands in bucket 1, the next weakest x_2 percent of hands in bucket 2, etc. The $x_1 ... x_n$ percentiles are generated by an exponential function,

such that the stronger the bucket, the fewer individual hands that comprise it.

3 Network Structure

For the remainder of the paper, discussion will be focused on belief prediction, which must be in place before action prediction can occur.

3.1 Action → Belief Edges

An alternative network could be formed by connecting past actions directly to the belief node to be predicted (instead of connecting the rounds via belief nodes). This approach was rejected due to the resulting explosion in belief node CPT size and its inability to incorporate information about the board cards.

3.2 Belief → Belief Edges

As it stands, however, the CPT size of the belief nodes is still out of control. Consider: for round-4, the number of possible round-3 buckets and distinct five card boards alone is $15 * \mathbb{C}(52, 5) \approx 40,000,000$. However, it is not the combination itself that is meaningful, but rather what the combination implies about the distribution over the next belief bucket. (For an example as to why these distributions are not all the same, consider the following: You are holding [Ah,Ac], and someone makes a very large bet on [5c,6c,7h,6h]. This should cause you more concern than the same action on [2c,5d,7h,Ts]. In the first case, it is much more likely for your opponent to have a hand beating your high pair, such as a straight, a flush, or three of a kind, than it is with the second board.)

If these distributions can be clustered into a sufficiently compressed set, the CPT size of the belief nodes will once again be tractable (see Section 5 for more detail). Before clustering can proceed, however, the belief transition probabilities need to be computed.

Computing Belief Transitions If a player is in bucket k at board b with hole cards c, and $b \rightarrow b'$ as a new card is revealed, the prior probability that the player is now in bucket k' is:

$$\mathbb{P}(k'|k, b, b') = \sum_{c \in C} \mathbb{P}(k'|c, k, b', b)\mathbb{P}(c|k, b', b)$$

Applying Bayes rule to the second term:

$$\mathbb{P}(k'|k, b, b') = \sum_{c \in C} \mathbb{P}(k'|c, k, b', b)\frac{\mathbb{P}(k|c, b', b)\mathbb{P}(c|b', b)}{\mathbb{P}(k|b', b)}$$

The next step invokes d-separation, a graph theoretical condition that is equivalent to conditional independence and can be used to simplify probabilistic expressions (Geiger, Verma, and Pearl 1990). Simplifying via Figure 3 yields:

$$\mathbb{P}(k'|k, b, b') = \sum_{c \in C} \mathbb{P}(k'|c, b')\frac{\mathbb{P}(k|c, b)\mathbb{P}(c|b')}{\mathbb{P}(k|b)} \quad (1)$$

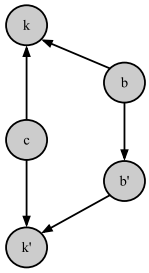

Figure 3: The relationships of the board, hole cards, and buckets between rounds

$\mathbb{P}(k|c, b)$ captures the membership of the hole card and board combination in bucket k (counter-intuitively not just 1 or 0, see (Kleij 2010) for the discussion of soft-bucketing). These probabilities are pre-computed for every possible c, b. This computation is made tractable by exploiting suit-symmetry, mentioned in Section 2.2. $\mathbb{P}(c|b')$ is simply one over the number of possible valid hole cards assignments given the cards not already in b', and $\mathbb{P}(k|b)$ is the prior probability of a pair of hole cards being in bucket k (defined during the processes of percentile bucketing).

Clustering Belief Transitions Many of these bucket-to-bucket transition profiles will look very similar (e.g. [2h,6c,6d,Ts] will behave almost identically to [2h,6c,6d,Js]), meaning each group could be accurately represented by a single distribution.

Such a clustering needs a distance metric between distributions to be defined. The most appropriate choice is the Earth Mover's Distance (EMD), which treats each distribution as piling of earth and the distance between two as the minimum amount of work necessary to transform one pile into another. This amounts to solving an instance of minimum-cost flow problem, where the costs of transferring probability mass between buckets can be bespoke.

To set the edge costs for the current application, remember that all hole card pairs are ranked according to their $E[S^2]$ value for the given board, and that the buckets partitioning them are not of uniform size. Instead, the bucket sizes decay at an exponential rate, meaning the average distance between a hole card pair in bucket i and a hole card pair in bucket $i + k$ is larger than the average distance between buckets j and $j + k$, if $j > i$. Figure 4 illustrates.

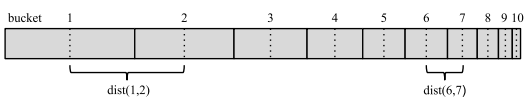

Figure 4: Distance between buckets

The cost of moving probability between buckets,

should therefore reflect the distance between their centers.

4 Inference

Given the trained belief network and some observed actions, we would like to make predictions about the associated belief sequences.

4.1 Derivation

To make this precise, the following notation is introduced:

- $\mathbf{B}$ is the set of boards encountered at each round: $\{b_2, b_3, b_4\}$
- $\mathbf{A_i}$ is the set of sets: $\{A_1, A_2, ..., A_i\}$
- A_i is the set of four individual actions taken on round i: $\{a_{1i}, a_{2i}, a_{3i}, a_{4i}\}$
- k_i is the belief-bucket pair (k_{1i}, k_{2i}) of player1 and player2 on round i

If we wish to predict the buckets for round i after all actions have been taken, we wish to compute:

$$\mathbb{P}(k_i, ..., k_1 | \mathbf{B}, \mathbf{A_i})$$

Expanding via the chain rule, and canceling by d-separation:

$$\mathbb{P}(k_i, ..., k_1 | \mathbf{B}, \mathbf{A}) = \mathbb{P}(k_1 | A_1) \prod_{j=2}^{i} \mathbb{P}(k_j | k_{j-1}, b_j, A_j) \tag{2}$$

Each term in this equation is available for lookup in the CPTs. The more complicated case is when no actions have been taken for the round we are interested in:

$$\mathbb{P}(k_i, ..., k_1 | \mathbf{B}, \mathbf{A_{i-1}})$$

$$= \mathbb{P}(k_1 | A_1) \left(\prod_{j=2}^{i-1} \mathbb{P}(k_j | k_{j-1}, b_j, A_j) \right) \mathbb{P}(k_i | k_{i-1}, b_i) \tag{3}$$

To compute $\mathbb{P}(k_i | k_{i-1}, b_i)$ would require marginalizing over all $(n_r + 4)^4$ possibilities of A_i (this is not quite right, see Section 5, but the general point remains). Since MCTS will be making this inference thousands and thousands of times, speed is of the essence. For this reason, it is prudent to replace the $\mathbb{P}(k_i | k_{i-1}, b_i)$ learned by the network and instead use the *a priori* version precomputed in Section 3.2.

With all relevant terms reduced to lookups, it is possible to identify the belief sequence most likely to be associated with the observed actions. Therefore, after all actions have been taken, the likelihood of each round-4 belief pair can be expressed as:

$$\mathbb{P}(k_4 | \mathbf{B}, \mathbf{A}) = \sum_{k_3, k_2, k_1} \mathbb{P}(k_4, k_3, k_2, k_1 | \mathbf{B}, \mathbf{A}) \tag{4}$$

4.2 Tractability

Obtaining a distribution over k_4, however, means computing (2) for the $10^2 * 20^2 * 15^2 * 10^2 \approx 10^9$ different belief assignments. Considering this computation needs to be as fast as possible for MCTS to take a sufficient number of samples, doing the full inference is out of the question.

One solution is to compute the probability of a belief assignment round by round, maintaining a reduced set of the m most likely hypotheses at each step. More specifically, for a given hand, start by computing $\mathbb{P}(k_1 | A_1)$ for each possible k_1, and keep only the m most likely assignments. Next, compute $\mathbb{P}(k_2 | k_1, b_2, A_2) \mathbb{P}(k_1 | A_1)$ using only the m assignments to k_1 retained from the previous step. Once again, throw away all but the m most likely k_2, k_1 assignments. Continuing this process will yield $\mathbb{P}(k_4, k_3, k_2, k_1 | \mathbf{B}, \mathbf{A})$ defined over a greatly reduced set of k_3, k_2, k_1 assignments. In this manner, (2) must only be computed $10^2 + m * 20^2 + m * 15^2 + m * 10^2 < 825 * m$ times.

5 Training

Past hand histories are first fitted to the described node abstractions. It should be mentioned here that the belief-transition clustering process has changed the network. Rather than belief nodes having the previous belief node and current board as parents (as depicted in Figure 1), they now have a single parent, whose value is the cluster number assigned to the distribution implied by their combination (see Section 3.2).

To learn predict round-4 actions, the networks in Figures 1 and 2 are trained via EM on both labeled and unlabeled data.

To learn a network capable of predicting showdown buckets, only the network in Figure 1 is trained via a simple maximum-likelihood computation exclusively on labeled data.

5.1 Data

The data itself has been drawn from past tournaments of the Annual Computer Poker Competition. Since the network's action prediction accuracy is so intimately tied with its bucket prediction accuracy, knowing the true labels for all hands is crucial for analysis. As human-generated data lacks complete hole card labeling, computer-generated data must be used.

5.2 CPT Size vs Training Data

In Section 3.2, it was mentioned that "sufficient compression" of the belief-transitions was necessary to achieve a managable CPT size. There are $\approx 10,000,000$ example games which reach the final round. We would like the CPT size for the belief nodes to be significantly smaller than this. In other words, if R is the number of different action sequences that can be taken in each round, and the number of belief-transition clusters is C, we would like: $R * C \ll 10^7$.

It would appear that $R = (n_r+4)^4$, but this is not so. Many sequences, such as "bet, call, fold, dummy" and "dummy, raise 1/2 pot, fold, fold" are simply nonsensical. The true value of R is $2 * (n_r + 4)^3 + 4 * (n_r + 4)^2 + 4 * (n_r + 4) + 1$.

Therefore, if $n_r = 6$, $R = 2,441$, meaning we would like to form less than $10^7/2,441 = 4,097$ clusters.

6 Evaluation

The accuracy of the hole card distribution: $P(c|a_1...a_n, P)$ must be evaluated.

6.1 Evaluating the hole card distribution

When MCTS simulates a game to showdown, a distribution over each player's hole cards is necessary to estimate the rewards for back-propagation. The parameters of the network in Figure 1, if trained exclusively on showdown hands, can be set by a simple maximum likelihood computation. The inference $P(k|a_1...a_n, \mathbf{B})$ can then made on a subset of these hands set aside for testing.

Bucket Accuracy The measure of accuracy used is obtained by sampling this belief-bucket distribution s times and computing the average distance between the predicted buckets $\hat{k_i}$ and the true bucket k^*.

$$D = \frac{\sum_{i=1}^{s} dist(\hat{k_i}, k^*)}{s}$$

The first choice for the distance function might be $\hat{k} - k^*$, but such a distance is ignorant of percentile bucketing. As can been seen in Figure 4, predicting $\hat{k} = 2$ when the $k^* = 1$ is therefore more inaccurate than predicting $\hat{k} = 7$ when $k^* = 6$. Again, the distance between bucket centers is used.

Showdown Accuracy While the above may be a good measure of general bucket accuracy on an arbitrary round, another method is needed for predicting the outcome of a round-4 showdown, in which all that matters is the relative ordering of the predicted buckets. Each player's inferred bucket distribution is sampled s times. The estimated probability of player1 winning the hand (W) is then:

$$\mathbb{P}(W) = \frac{\sum_{j=1}^{s} M(\widehat{k_{1,j}}, \widehat{k_{2,j}})}{s}$$

$$M(\hat{k_1}, \hat{k_2}) = \begin{cases} 1 & \text{if } \hat{k_1} > \hat{k_2} \\ 0 & \text{if } \hat{k_1} < \hat{k_2} \\ .5 & \text{if } \hat{k_1} = \hat{k_2} \end{cases}$$

The probability of predicting the correct winner for each test example e is then computed using the true bucket labels k^*. Each probability is weighted by the amount of money exchanged a at the hand's conclusion. Getting big-money hands wrong translates into back-propagating a larger error through the MCTS algorithm, so the measure of showdown accuracy A should be correspondingly weighted.

$$A = \frac{\sum_{e=1}^{n} F(\mathbb{P}(W_e), k^*_{1,e}, k^*_{2,e}) * a_e}{n}$$

$$F(p, k^*_1, k^*_2) = \begin{cases} p & \text{if } k^*_1 > k^*_2 \\ 1-p & \text{if } k^*_1 < k^*_2 \\ .5 & \text{if } k^*_1 = k^*_2 \end{cases}$$

7 Progress and Future Work

All the terms necessary for computing the inference in (4.1) are available for lookup. The only uncompleted task impeding the training and testing of the networks is the clustering of belief-transitions. Even with parallelized code, computing the full distance matrix for the round-3 $\rightarrow$ round-4 transitions is estimated to take ≈ 600 days, so a new approach is obviously needed.

Also missing is a measure of accuracy for action prediction.

Once both networks are trained and tested, the final step will be to form a fully functional player by integrating the networks with MCTS.

References

Geiger, D.; Verma, T.; and Pearl, J. 1990. Identifying independence in bayesian networks. *Networks* 20(5):507–534.

Johanson, M.; Zinkevich, M.; and Bowling, M. 2007. Computing robust counter-strategies. *Advances in neural information processing systems* 20:721–728.

Kleij, A. 2010. Monte carlo tree search and opponent modeling through player clustering in no-limit texas hold'em poker. *M. Sc. University of Groningen, Netherlands*.

Ponsen, M.; Gerritsen, G.; and Chaslot, G. 2010. Integrating opponent models with monte-carlo tree search in poker. In *Proc. Conf. Assoc. Adv. Artif. Intell.: Inter. Decis. Theory Game Theory Workshop, Atlanta, Georgia*, 37–42.

Van den Broeck, G.; Driessens, K.; and Ramon, J. 2009. Monte-carlo tree search in poker using expected reward distributions. In *Advances in Machine Learning*. Springer. 367–381.