EFFICIENT APPROXIMATION TECHNIQUES IN BAYESIAN BELIEF NETWORKS FOR ONLINE POKER

(draft)

by

Nathan Matthews

Temple University, 2005

Introduction

The purpose of this project is to develop one or more Bayesian Networks (also called Causal Networks, Belief Networks, or Directed Markov Fields) to be able to play online poker. The method used could apply to many card games or games in general, but for this project the network was tuned for Texas Hold'em (limit games). It will be shown that proper application of these networks to online poker games (such as partypoker.com or pokertable.com) can produce net-winning strategy, even when many variables at the poker table are hidden. First, provided is an outline of the style of poker used, and second is an overview of bayesian network structure and teminology.

Texas Hold'em

In this game, each player at the table holds two cards (called the hole cards) which only he sees. There are four stages to the game. Initially, every player is dealt hole cards, and each player is allowed to bet (two players, called the blinds, must bet this turn). This is called the pre-flop turn. Next, three cards are placed on the table. These are communal cards, which can be used by any player to complete a hand. Players are then allowed to bet again. After this, another single card is added to the table, which is called the turn. After more betting, the last card (called the river) is placed, and a final round of betting, after which hands are shown. For every player, any combination of five out of his seven visible cards will make up his hand.

To actually play this game well, there are many intricacies and strategies, but basic strategy can be summed up easily. There are three major factors a player considers to determine his standing at the table: what cards his opponents likely hold (determined by watching their actions), hands the player could draw and their likelihood (called pot odds), and the amount of the pot compared to the current bet amount. After the player guesses his most likely standing, he determines whether it is worth putting money into the pot (and in certain circumstances, performs more intricate strategic maneuvers).

There are many aspects of poker games which make them likely candidates for bayesian

networks.

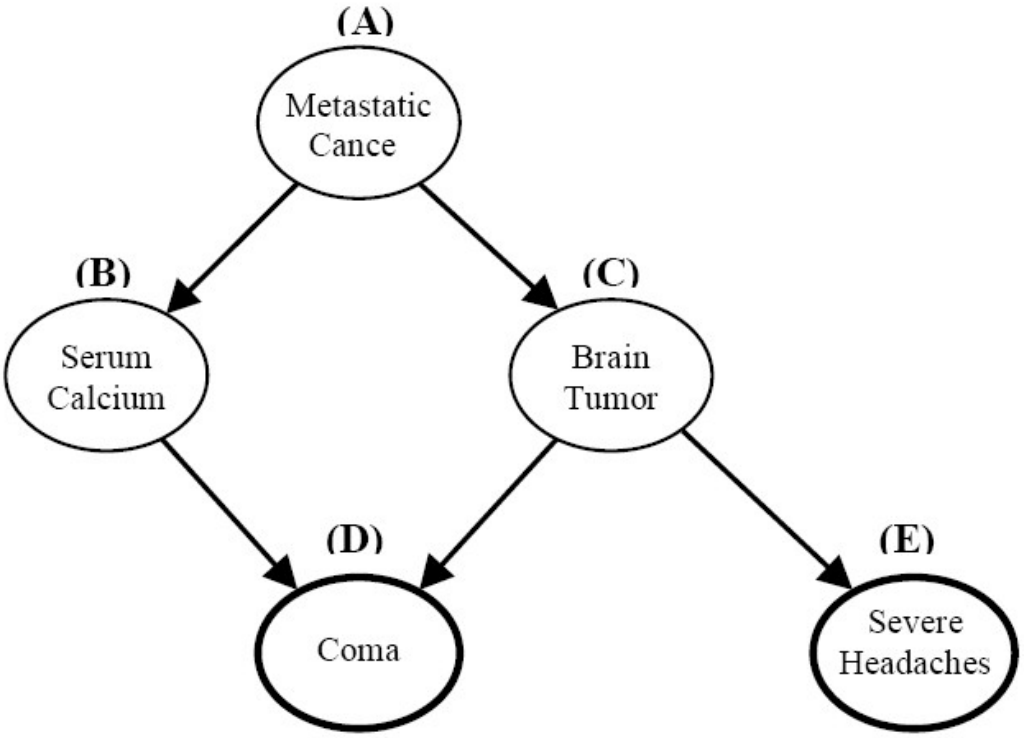
First, one of the major niches in AI filled by Bayesian networks is reasoning with uncertainty. Bayesian networks are able to produce good guesses even when the full state of the problem space is not known, and when only partial causal information is present to deal with it. Uncertainty is the main characteristic of poker, and the human mind already performs simple probabilistic leaps when playing the game which translate into causal information.

Second, while there is a great deal of formal poker strategy in books and tutorials, most of this information is a "micro-managed" approach: it gives advice for specific instances in poker, but not overall consideration. It is difficult to take the thousands of small rules given for poker, and unify them into one algorithm.

Third, and most convincingly, given a player is able to decide correctly what to do if he knows his standing at the table, it is more difficult to guess what position the opponents are in. However, if the player can make the assumption that his opponents are rational players operating in much the same way he does, he can "reverse-engineer" their *actions* at the poker table into the *reasons* they were likely to have made those choices. It is possible for human players to do this operating intuitively, but not with great accuracy, meaning human players must make more conservative estimates than are necessary. This kind of "reverse inference" is very easy for a Bayesian Network to do.

Bayesian Networks

A Bayesian network consists basically of two parts: a qualitative part, which describes causal relationships among random variables, and a quantitative part which describes the conditional probabilities of those variables given their conditioning parents. The state of a certain problem is described by a set of variables, each of which can be in one or more states. In normal practice these states are discreet, although it is possible to have continuous variables in some methods. For example, consider the following network (Fig. 1):



$\Pr(A)$	
a_1	0.2
a_2	0.8

$\Pr(B\|A)$	a_1	a_2
b_1	0.8	0.2
b_2	0.2	0.8

$\Pr(C\|A)$	a_1	a_2
c_1	0.2	0.05
c_2	0.8	0.95

$\Pr(E\|C)$	c_1	c_2
e_1	0.8	0.6
e_2	0.2	0.4

$\Pr(D\|B,C)$	b_1		b_2	
	c_1	c_2	c_1	c_2
d_1	0.80	0.80	0.80	0.05
d_2	0.20	0.20	0.20	0.95

Fig. 1 – the "Coma" network [Cheng 01]

In this network we see unknown variables A – E, each of which has two states (indicating the named symptom/disease is present or absent, respectively). The graph is the qualitative part and the tables are the quantitative part. Pr(A) represents the prior probability of the variable A being in state "true". That is, given no evidence, A is in state a1 with 20% probability and in state a2 with 80%

probability. Pr(B|A) is the conditional probability of B, given A. Therefore if A is in state a1, B has an 80% chance of being in state b1; else, it has a 20% chance. Pr(D|B,C) indicates the conditional probability of D given both its conditioning parents (since it has two converting arrows in the graph, from B and C). D equals d1 with 80% probability unless B is b2 and C is c2, in which case D is only d1 with 5% probability.

The total joint probability of the network can be taken by multiplying the probability distributions of each variable conditional on its parents. If **X** is the set of all variables in the network, and Pa(X**)** represents the parent variables of X, then the joint probability of **X** is:

$$\Pr(\mathbf{X}) = \prod_{i=1}^{n} \Pr(X_i | Pa(X_i))$$

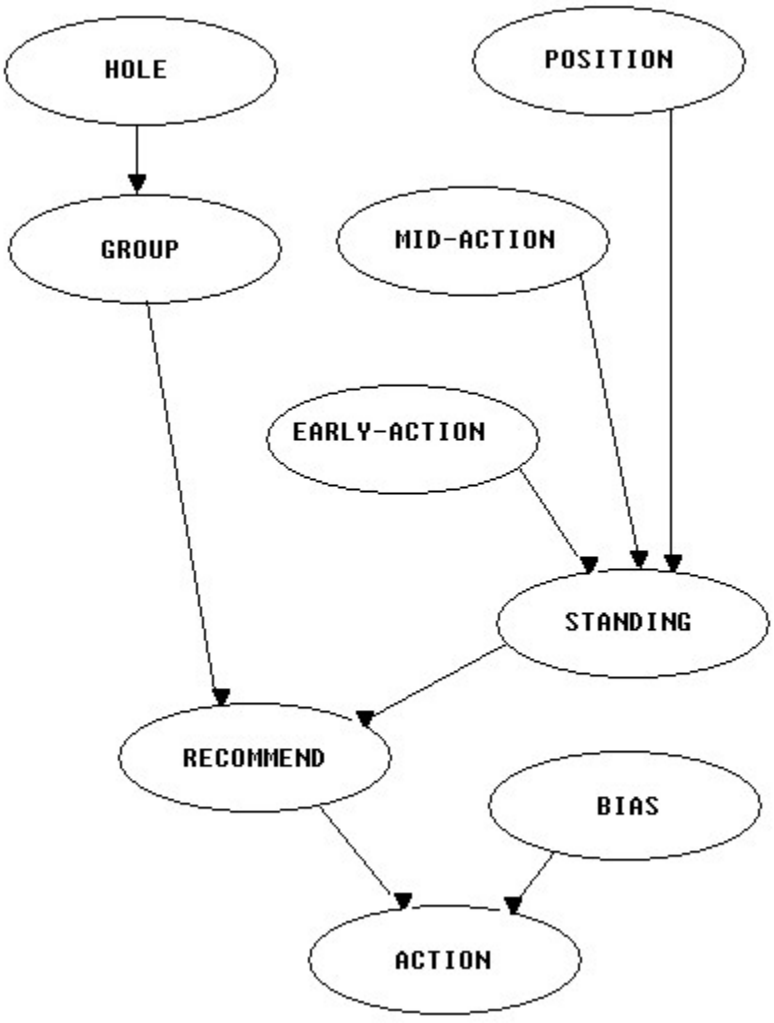
Some of the variables, called evidence variables, are in a known state and are summed out of the distribution after finding the joint probability, to find total probability conditional on the evidence. In practice, since the resulting distribution is huge for a moderate number of variables, in order to find out information about the posterior distribution of a "query" variable (its marginal probability given its conditioning variables), alternative inference methods exist to limit the number of multiplications in the process. Algorithms such as bucket tree, junction tree, conditional independence, and recursive decombination apply this method. However, the problem has been proven NP-hard, and for large networks is still prohibitively slow. After the network setup of the poker program is presented, the chosen solution to this problem is given.

Bayesian Networks for Poker: Pre-Flop

When adapting the game of poker to Bayesian Networks, it is clear the the pre-flop stage is distinct from the three post-flop stages. For one, early position players (those to act first) know almost nothing about their opponents in this round, and must react strictly according to their hole cards. Also,

there is little information for any player about draw probabilities, making the post-flop computation of pot odds impractical. Players must also in general play much tighter pre-flop (that is, take less risk, as opposed to playing loose). The three post-flop rounds are almost identical in the mechanics of deciding actions, differing only in probabilities and strategies. They are all represented by the same network.

The pre-flop network was designed as follows.



Briefly, the possible states of these variables are:

HOLE: pairs (AA, QQ, 55, etc.), unsuited (AK, 37, etc.), and suited.

GROUP: G1-G6. power-groupings of hole cards into 6 groups

POSITION: small blind (SB), big blind (BB), under-the-gun (UTG), early (E), mid (M), late(L)

EARLY-ACTION: FOLD, CALL, RAISE (actions opponents in early position took)

MID-ACTION: FOLD, CALL, RAISE (actions opponents in middle position took)

STANDING: combinations of G1-G6 and F,C,R (group required to not fold, and action)

RECOMMEND: FOLD, CALL, RAISE (the action to take playing straight)

BIAS: LOOSE, NORMAL, TIGHT (player style)

ACTION: FOLD, CALL, RAISE (actual action taken)

The causal relationships can be explained: what hole card is held (a pair, a suited non-pair, or an unsuited non-pair) determined what "power-group" those cards are in (according to six sub-regions of a graph of power rankings for all possible hole combinations). AA is the highest, and 27 unsuited is the lowest. These power rankings are determined experimentally by seeing how often they draw powerful hands in trial games where no player folds, and weighting the outcomes. The prior probability of the HOLE variable assigns a .45% probability to each pair, .3% to each suited non-pair, and .9% to each non-suited non-pair. The hole cards directly cause the power of the group.

The position variable simply notes where at the table the player sits. The early and mid-action nodes indicate what the highest aggression was in different parts of the table (aggression at early parts is more dangerous, because those players are taking a bigger risk). The combination of position and activity at the table indications the STANDING. In practice this node is two nodes: one which indicates what power group should be required to not fold against that activity at that position, and the other to indicate what action to take (call or raise) if the player does not fold.

Based on the actual power group the player has, and the required group and action from the

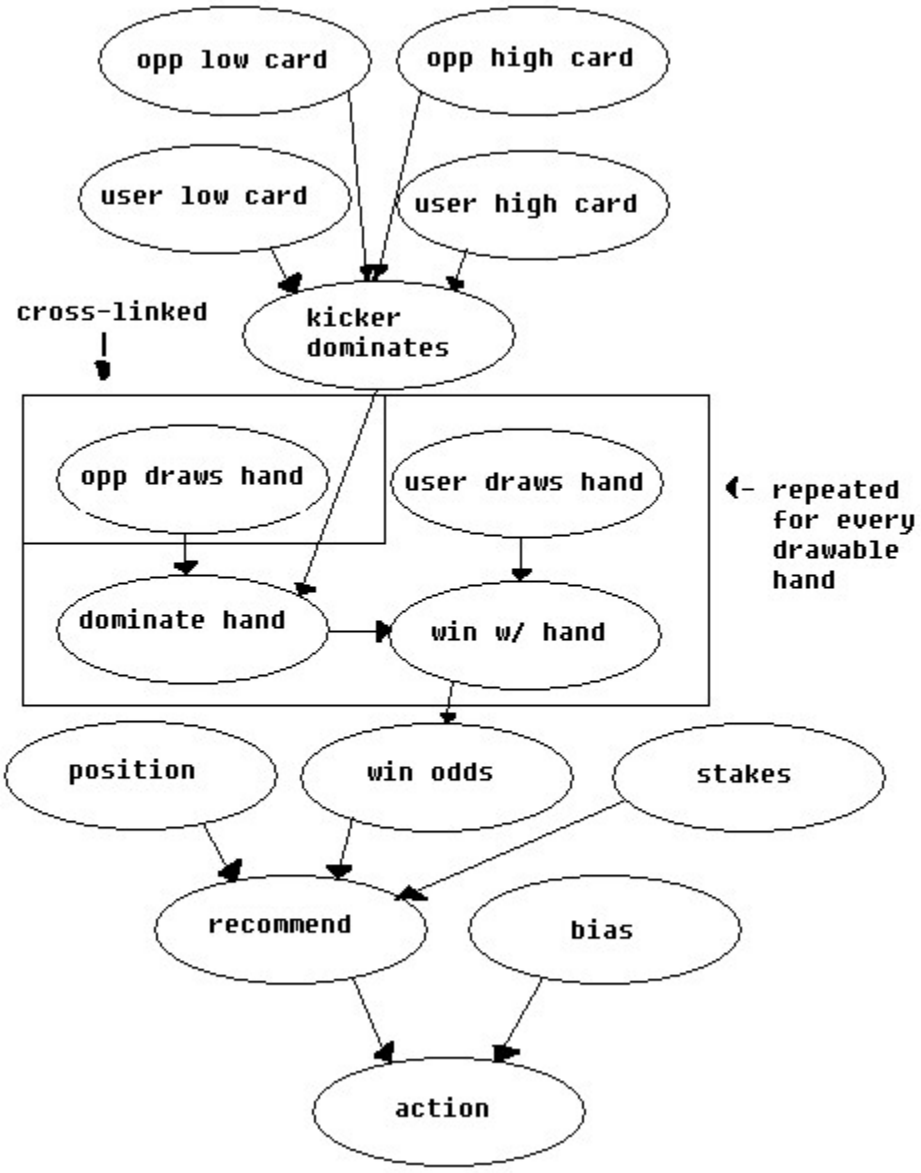
STANDING, the RECOMMEND node indicates what the best move would be. Then, to indicate that some players play looser or tighter, the BIAS node in addition decides the actual ACTION the player should take.

As indicated earlier, the power of a Bayesian network is that it can perform inference from prior evidence, or from "posterior" evidence: that is, even though for the player, inference proceeds as in the previous two paragraphs, we can reverse the process. Given what action a player DID take, simulation can determine the likely distribution of other variables in the network. Therefore, there are two ways the network is used. When it is the player's turn to act, the evidence variables are HOLE, POSITION, EARLY-ACTION, and MID-ACTION. These are things the player observes about his own state. The player then queries the ACTION variable to determine what to do.

When each of the opponents acts, however, the evidence nodes are POSITION, EARLY-ACTION, MID-ACTION, and ACTION; these are the only things known in the player's observation of his opponent. From there, the GROUP variable can be queried to determine the likely power group that opponent holds. Although this is not really useful in the pre-flop round (because aggression itself is a better indicator), this information is used externally to help calculate the opponents' draw probabilities in the post-flop rounds, as will be seen.

Post-Flop

The post-flop network is different from pre-flop primarily in that it relies more directly on the estimated probability of winning certain hands, and less on intuitive understanding of aggression (which is still a factor). The network's structure is more complicated, however. Each possible hand must have an independent draw probability, the maximum opponent's draw probability, the probability of dominating each opponent hand with each user hand, resulting in the probability of winning on every possible hand, producing the total probability of being the strongest draw on the table. The network looks like this:



Notice that variables within boxes are repeated for every possible hand; links in and out of the boxes (i.e, from “kicker dominates” and to “win odds” are duplicated, and each “opp draws” is cross-linked to each “dominate hand”). This cross-linking produces an interesting effect which is discussed later. Briefly, the possible states of the variables are:

OPP LOW CARD: 2-A (the low card in the draw, or the kicker for a pair)

OPP HIGH CARD: the top card of the draw (like A for royal flush)

USER LOW/HIGH CARD: same as OPP, for user

KICKER DOMINATES: False or True (whether same hand is stronger for user)

OPP DRAWS HAND X: False or True (probability *some* opponent will draw hand X)

USER DRAWS HAND X: False or True (probability player will draw hand X)

WIN WITH HAND X: False or True (probability player will win with hand X)

WIN ODDS: False or True (probability player will not be beaten by any hand)

POSITION: under-the-gun (UTG), early (E), mid (M), late (M) (player position)

STAKES: Low or High (inverse of pot odds, indicating probability of a *low* pot/bet ratio)

RECOMMEND/BIAS/ACTION: as in pre-flop

For the player's turn, evidence about opponent draw probabilities are obtained externally from three sources for the under-the-gun player, and four thereafter. First, the power group distribution determined from the pre-flop round is applied, and second, this distribution is modified by counting outs (unknown cards which could complete the player's hand), and then this distribution is further modified to reflect the total probability of drawing certain hands in general (for instance, the extreme unlikelihood of a royal flush). For the under-the-gun player, this is all the information available, and the prior probabilities for all the opponent hand draw variables are mostly uniform in the first two sources. However, for every opponent action, the network's query and evidence nodes are reversed as for the pre-flop round, to determine what the opponent players *believe* to be their total win probability. A further inference (based on the overall likelihood of drawing hands) determines the likelihood that that player will draw each hand in the future, based on their implicit belief in this. (Of course, as before, the assumption is that the player is playing against rational opponents, outside of the supplied bias.) The maximum probability that *some* player has to draw a hand is used as the player's evidence for the "opponent draws" variables.

Once these variables are combined to form the total probability of dominating the table, the next two evidence nodes, stakes and position, determine the recommended action. In general, in the post-flop rounds, folding occurs when the size of the pot does not warrant the risk of a bet. For

instance, if the pot is $40 and the bet is $5, the pot odds are 40:5 or 8:1. Therefore, the player need only win this pot one out of every eight times to break even, or more to profit. Therefore, a win probability of *less* than 1 in 8 indicates a fold. In addition, the position variable indicates greater uncertainty in early position because no action has been seen yet, and riskier bets are avoided. The combined probability of making a profit on this round (in addition to strategy relating to the position of the player) will determine the recommended action, which is modified by bias as before to produce the actual action.

Practical Considerations

The structure of these two networks guided the search for an appropriate algorithm to query the network during gameplay. First, it must be easy to have evidence nodes either at the "top" or the "bottom" of the network, and to perform equally well. Second, since online poker usually proceeds very quickly (a second for most opponents to act), the network evaluation must be very fast.

Solutions

There are a variety of simulations and approximations to solving Bayesian networks which produce adequately precise results (under certain conditions). For an initial attempt, the poker networks were approximated using a quasi-Monte Carlo technique called Logic Sampling using Markov Blankets. In this method, each non-evidence variable is initialized to a random state. Then for a certain number of iterations, each variable in topological order evaluates is posterior distribution based on its Markov blanket, that is, the variables in the network which make it conditionally independent from the rest of the graph: it's parents, children, and childrens' parents, i.e. the distribution

$$Pr(X) = Pr(X|Pa(X)) * Pr(Ch(X) | X, Sp(X))$$

Where Pa(X) are the parents of X, Ch(X) are the children of X, and Sp(X) are the "spouses" of X, the parents of X's children. The variable then reinstantiates itself to a new state randomly, such that the

outcome is proportional to the distribution. Each iteration of this method is a sample of the total joint probability, and because the variable distributions are used to select each successive sample (making it a quasi-Monte Carlo method), the result is an average distribution of states for each variable, given the fixed evidence nodes.

As is noted in the literature and confirmed by tests, this simple method of logic sampling suffers from a lack of adequate sample distribution. Evidence with very low probability in the network will tend to never be sampled, skewing the resulting joint distributions. In the case of the pre-flop network, every single state in the HOLE variable has a very low probability, making this method highly inaccurate in practice. In tests of simple networks of only five nodes which included low probability conditional distributions, error of up to 30% with 10,000 iterations was observed. Error of even a fraction of that in a poker game is enough to turn steady profit into steady loss.

There are many methods available which can help correct the sampling problem: stratified sampling, importance weighting, and latin-hypercube sampling, to name just some. Latin hypercube sampling, in particular, has shown to improve the sampling accuracy of other schemes. {DRAFT: not implemented}

Future Work

Because of the limitations of approximations in networks containing very low probabilities, it may be more feasible to implement a direct solution using some form of graph separation. In particular, bucket trees are a relatively simple method with good results. This method uses a good ordering of the variables in the network (which is unfortunately NP-complete to find in the general case) to reduce the overall number of multiplications and also the storage space of the algorithm. The inherent difficulty of using this method with the poker-networks is the post-flop network, in which there is cross-linking between the “opponent draw” and “dominate hand” variables for each hand. In preliminary tests this section of the network was a drawback to the bucket-tree method and exploded

the joint probability too quickly.

Another possible method is the use of Query-DAGS, which are an analagous network made up of arithmetic operations which are computed using a normal Bayesian network solution. The advantage of this network is that changes in evidence nodes can be propagated through the network without involving many of the other variables. Since the same network is used in the three post-flop rounds, this could greatly reduce the overall computation time, although space guarantees for the QDAG algorithm can be no better the the guarantees provided by the method which generated them, in this case, bucket-trees.

Conclusions

{DRAFT: implement btree and lhcube first}

Bibliography

{DRAFT}