Unified Tree Search for Evaluating Gambling Games

J. Mark Ettinger
jettinger35@gmail.com

January 10, 2016

1 Motivation

This essay considers the automatic evaluation of games against nature using combined methods of exact inference, sampling, and Monte Carlo Tree Search (MCTS) incorporating Bayesian methods. The motivation for these investigations is the goal of automatically (i.e. with minimal human intervention) estimating

$$\mu = 1 + \frac{E(X)}{E(B)}$$

and

$$\sigma^2 = E((1 + \frac{X}{B} - \mu)^2)$$

and in some cases, where it is computationally feasible, the full distribution for

$$1 + \frac{X}{B}.$$

where (B, X) are jointly distributed random variables representing the bet and profit of a wager in games of chance. μ is usually called the *Return To Player (RTP)*. A related quantity, the *house advantage* is defined as

$$1 - \mu = -\frac{E(X)}{E(B)}.$$

Of particular interest and difficulty are games of chance incorporating *strategy*, e.g. Blackjack, Video Poker, etc. Unlike games of pure chance, for example Roulette or slot machines, where random strategies are optimal, in strategic games of chance it is necessary to estimate μ and σ^2 utilizing optimal or near optimal strategies (hereafter called *good strategies*) or the estimates will be inadequate for assessing risk. Finding good strategies is a challenging task closely related to general game-playing and other game-theoretic problems in artificial intelligence. The essential challenge is efficient search of the game decision tree, balancing exploration and exploitation. Probabilistic inference in other settings, e.g. Bayesian nets, is $\#P$-complete and while we are not aware of specific complexity results for our more restricted problem concerning trees, we conjecture that a formal definition of the problem may be just as difficult.

2 Estimations Using the Hoeffding Bound

For games of pure chance we may estimate μ by sampling outcomes using random strategies, which are optimal in this case, and using Hoeffding's bound applied to the samples. For strategic games of chance we may use the UCT algorithm (*Upper Confidence Bound for Trees*), a form of Monte Carlo Tree Search, to search for good strategies and then using a good strategy to sample outcomes, use Hoeffding's bound to estimate μ. Hoeffding's bound is

$$Prob(|\bar{X} - E[\bar{X}]| \geq t) \leq 2\mathrm{exp}\left(-\frac{2n^2t^2}{\sum(b_i - a_i)^2}\right).$$

One advantage of this technique is that we estimate μ directly, rather than first estimating the entire distribution for the game and then calculating μ based on this distribution as we do in the algorithm described below. Unfortunately, many games of chance have very large, very rare payouts which makes the denominator in the exponential very large. This implies that for many games an intractable number of samples (n) are required to yield a tight bound. This motivates the incorporation of Bayesian techniques to decrease the required number of samples by leveraging prior information about the game.

3 The Goal of the Algorithm

For a given game state, s, our goal is to estimate the *value* $V(s)$ of the state. In the most general case the value of a state is a Dirichlet distribution over distributions over game outcomes. We view a categorical distribution as being a limiting case of a Dirichlet (i.e. a point mass) and a numerical value as being a special case of a categorical distribution (i.e. where a single outcome has probability equal to one). Dirichlet distributions arise in the present context as posterior distributions incorporating the results of sampling via search in the game tree. Categorical distributions arise in two ways, either as a marginal distribution of a Dirichlet in the context of sampling or directly in the case of exhaustive and exact inference. Once we have a posterior distribution for $V(s)$ we can obtain either point estimates or posterior distributions for the primary quantities of interest, μ and σ^2.

4 Algorithmic Approach

As our goal is to estimate $V(s)$, we proceed recursively, first estimating $V(s')$ for some (in the case of sample-based methods) or all (in the case of exact inference or sample-based methods) *child states* s' of s and then using the child estimates to estimate $V(s)$. A *child state* is a game state that is reachable by an immediate legal move of the game. An important aspect of our algorithm is that the method for estimating the value of different states in the game tree is allowed to vary from state to state. For some states we may insist on exact inference where for other states we may prefer a sampling-based method. The valuation method for a given state in the tree is specified by a function as discussed below.

We differentiate between *nature states* where it is nature's turn to move and *player states* where it is the player's turn to move. This distinction is important because the value of a state as a function of the value of child states differs markedly between the two types of states. The value of a nature state, where the moves are stochastic, is essentially a probabilistic weighted average of the value of the child states. The value of a player state is essentially a maximum over the values of child states. This is discussed in greater detail below.

For a given game of interest, G, we assume a state class is given which describes the state of the game and a function *method()*, which is a function from states to a set of methods of evaluation, e.g. {*exact, randomSample, Bayesian Monte Carlo Tree Search (BMCTS)* }, and relevant parameters. For example a computational budget parameter is required if the method is sample-based. The function *method()* allows a search of the game tree where the method of evaluating a state can vary depending on various factors, including sample complexity, desired accuracy, etc.

4.1 Nature States

Since nature moves stochastically, there are two methods of valuation: exact inference and random sampling. The essential challenge for nature states is to calculate or estimate the transition probabilities to child states. If these probabilities can be computed analytically then the game state class must define member function getMoves() which should return a list of pairs $(p_i, {s_i}')$ where p_i is the probability of transitioning to state ${s_i}'$. If the transition probabilities are to be estimated by sampling, the game state class must define member function getPrior() which returns an appropriate prior distribution. Transitions are randomly sampled and the prior is updated in the usual way by Bayes' rule. By recursion, we assume we have calculated or estimated the values of the child states.

If we estimate the values of the child states by sampling, the computational budget parameter provides a limit on the exploration effort we may distribute among the child states. This effort should be concentrated on the child Dirichlet distributions with the greatest "uncertainty", similar to the exploration/exploitation tradeoff addressed by Bayesian Monte Carlo Tree Search for player states as discussed below. Notice that in the context of games against nature, categorical distributions (i.e. point mass Dirichlet distributions) are considered to have zero uncertainty because of the inherent stochastic nature of games against nature. Uncertainty in games against nature only arises from imperfect knowledge *about* a categorical distribution, i.e a Dirichlet distribution. There are various ways of quantifying the "uncertainty" of a Dirichlet, including various matrix norms applied to the covariance matrix and a simple sum over the Dirichlet hyperparameter "pseduocounts"

As a function of the values of the child states, the value of the nature state is given by a mixture distribution over the child distributions with mixing coefficients given by the calculated or estimated p_i.

4.2 Player States

For player states s, $V(s)$ is a maximum over the values of child states. In other words, the player chooses the move for which the expected value of the child state is maximal. For exact inference, the value of the player state is the child categorical distribution with maximal expectation.

In the case of sampling, the value of the player state is the child Dirichlet distribution which has the maximal expectation obtained from the Dirichlet by marginalizing over the Dirichlet hyperparameter. For a sampling approach to valuing a player state, we use a form of Bayesian Monte Carlo Tree Search (BMCTS), inspired by UCT but incorporating prior information to increase efficiency. The fundamental idea of MCTS is to address the exploration/exploitation tradeoff by allocating sampling resources to child nodes that maximize a function of both expectation and uncertainty. To determine the next sample, UCT utilizes the formula

$$v_i + C\sqrt{\frac{N}{n_i}}.$$

where v_i is an estimate of the value of child state i, N is the number of samples originating from the player state, n_i is the number of samples from child i, and C is a constant which determines the relative importance of exploration and expoitation. BMCTS is similar in spirit but v_i is a Bayesian expected value rather than frequentist and, as discussed above in the nature state subsection, there are various possible ways for the exploration term to capture "uncertainty."

5 Conclusion

We have outlined an algorithm for estimating the value of states of one-player games against nature. This algorithm unifies various alternative solution methods as different methods can be applied to different states in the game tree as appropriate. It is also important to indicate that this algorithm is flexible with respect to the amount of human assistance that is desired. In particular, the choice of the set of initial states to analyze can have a dramatic impact on the overall complexity required for a complete game analysis. In some cases an exhaustive valuation over all possible initial states of nature will be computationally feasible. In other cases where this is combinatorially prohibitive, it may be possible for human analysis to reduce the size of the set of initial states by symmetry considerations. Therefore it is important to specify the initial state in order to permit an efficient solution. For example, if the analysis is designed such that the initial state is a nature state and if nature has a combinatorially large branching factor, the game may not appear to be tractable. On the other hand, if the analysis is designed so that the initial state is a player state and the set of initial states required to analyze the complete game is not unreasonably large, perhaps due to symmetry considerations, then the game analysis may be tractable.