How a Bayesian Approaches Games Like Chess

Eric B. Baum
NEC Research Institute, 4 Independence Way, Princeton NJ 08540
eric@research.NJ.NEC.COM

Abstract

The point of game tree search is to insulate oneself from errors in the evaluation function. The standard approach is to grow a full width tree as deep as time allows, and then value the tree as if the leaf evaluations were exact. This has been effective in many games because of the computational efficiency of the alpha-beta algorithm. A Bayesian would suggest instead to train a model of one's uncertainty. This model adds extra information in addition to the standard evaluation function. Within such a formal model, there is an optimal tree growth procedure and an optimal method of valuing the tree. We describe how to optimally value the tree, and how to approximate on line the optimal tree to search. Our tree growth procedure provably approximates the contribution of each leaf to the utility in the limit where we grow a large tree, taking explicit account of the interactions between expanding different leaves. Our algorithms run (under reasonable assumptions) in linear time and hence except for a small constant factor, are as efficient as alpha-beta.

Introduction

[Shannon, 1950] proposed that computers should play games like chess by growing a full width game tree as deeply as time permits, heuristically assigning a numerical evaluation to each leaf, propagating these numbers up the tree by minimax, and choosing as the “best move” the child of the root with the largest number. Now the whole point of search (as opposed to just picking whichever child looks best to an evaluation function) is to insulate oneself from errors in the evaluation function. When one searches below a node, one gains more information and one’s opinion of the value of that node may change. Such “opinion changes” are inherently probabilistic. They occur because one’s information or computational abilities are unable to distinguish different states, e.g. a node with a given set of features might have different values. In this paper we adopt a probabilistic model of opinion changes, describe how optimally to value the tree in this model, and give a linear time algorithm for growing approximately the most utilitarian tree.

We first argue that minimax, while producing best play in games between “perfect” players who can afford to search the entire game tree, is not the best way to utilize inexact leaf values in a given partial tree. Nor is another old idea [Pearl, 1984] that we call “naive probability update.” Instead, from the point of view of a Bayesian, one should model one’s uncertainty, and within the context of such a probabilistic model derive the optimal strategy. This leads to a propagation rule we call “best play for imperfect players,” or BPIP. BPIP has a simple recursive definition.

To implement BPIP, one needs a “probabilistic model of extra information.” The words “extra information” denote the idea that an imperfect game player, upon reaching a node in the game tree during later play in the game, will then be able to search deeper below that node, and thus gain access to more information than he previously had. We adopt an evaluation function which, rather than returning a single number estimating the expected value of further play in the game, also returns a probability distribution \( P_L(x) \) giving the likelihood of opinion changes in that number if the node were searched deeper. \( P_L(x) \) is the probability that if we expanded leaf \( L \) to some depth, the backed up value of leaf \( L \) would then be found to be \( x \). The mean of our distribution valued evaluation function is an ordinary evaluation function, but our distribution gives the probability of various deviations. We describe a statistical method for training such an evaluation function. In essence one may empirically measure the likelihood of various opinion changes as a function of various features.

We assume these distributions are independent.\(^2\) (Note: we are not assuming that the probabilities of winning at different nodes are independent, as have [Pearl, 1984; Chi & Nau, 1989]. We are assuming that the errors in our estimate of these probabilities are independent.)

\(^1\)This is a super-abbreviated discussion of [Baum and Smith, 1993] written by EBB for this conference.

\(^2\)We also make a “depth free” assumption, implicitly also made by minimax, and likely to be less important than the independence assumption.
Our "ESS" is a value $\psi_L$ associated with each leaf that provably approximates $\delta_L$ to within a factor of 2. Under some practical restrictions, described in the full paper, the $\psi_L$ is identically equal $\delta_L$. The ESS itself has intuitive meaning: it is the expected absolute change in $U$ when we expand leaf $L$. When you expand leaf $L$, the remaining utility from expanding
all the rest of the leaves can either rise or fall. When it falls, this takes you closer to moving, since remember the natural condition for terminating search and selecting a move is when the remaining utility falls below the time cost. When \( U \) rises, this means that the result from expanding leaf \( L \) was surprising, and that you were mistaken about the utility of expansion. Both types of information are valuable, and the ESS assigns them equal value\(^4\). Alternatively, the ESS can be seen as the best estimate of the \( a \) posteriori change in the expected utility. Thus the ESS is a natural leaf importance measure in its own right, and furthermore is shown by a (hard to prove) theorem to approximate the contribution to the utility made by the leaf in the large expansion limit. We have algorithms that compute the ESS values \( V_L \) for all leaves \( L \) in our search tree, exactly, in a number of arithmetic operations depending only linearly on the size of the tree.

This, then, is our proposal: valuate the tree using BPIP, and grow it by repeatedly expanding the leaves of our current tree which have the largest ESS values. Keep re-growing and re-valuating until the utility of further growth is smaller than the estimated time-cost it would take, then output the best move. Using various algorithmic devices we propose, in particular the "gulp trick," certain multilinearity lemmas, and our "influence function" methods, this entire move-finding procedure will run in time depending only linearly on the size of the final search tree (that is, after all growth is complete).

The constant factors in our time bounds are small. Thus for example, for chess, assume reasonably that the time to evaluate a position is large compared to the mean depth of search (times a small computation time), and that we tune our algorithm to search three times as deep along the lines judged most important as alpha-beta. Then if our algorithm and alpha-beta explore for the same amount of time, the tree we grow will be up to three times as deep, but contain about half as many leaves, as that of alpha-beta.

Previous work
A number of tree growth algorithms have previously been proposed, e.g. by [McAllester, 1988; Palay, 1985; Rivest, 1988; and Russell and Wefald, 1991]. From our point of view these authors were all (either implicitly or explicitly) striving to calculate the decision theoretic optimal leaf to expand next. Our approach of stating the Bayesian model of search, and then giving a provably efficient algorithm approximating Best Play for Imperfect Players can thus be seen as unifying, formalizing, and (at least from a theoretical point of view) to a certain extent resolving this line of research. Previous heuristic ideas like "singular extensions" [Anantharaman, Campbell, & Hsu, 1990; Anantharaman, 1990], and "quiescence search" [Beal, 1990] as well as alpha-beta style cutoffs occur automatically in our procedure. In contrast to the previous approaches, which were all greedy or ad hoc, our leaf importance measure accounts for the possible outcomes of future expansion. We review previous work in more detail in the full version. We also remark there that the algorithms of [McAllester, 1988] and [Russell & Wefald, 1991] take time superlinear in the number of leaves.

Palay was the first author to propose the use of distribution valued evaluation functions and also proposed the same equations for propagation of probability distributions that we do, but his use of them was approximate, and his motivation and application different. Our goal of optimal search, our use of a utility based stopping condition and of evaluation functions which return distributions were stimulated by Russell and Wefald, but we believe improve on theirs.

Pointer
We have provided this brief introduction as we are unable to coherently describe the details within the present page limitations. Details may be found in [Baum & Smith, 1993] which has been submitted for publication, and in the interim may be obtained by anonymous ftp from external.nj.nec.com, in file pub/eric/papers/game.ps.Z.

Experiments in progress will be reported elsewhere.

References
E.B. Baum, W.D. Smith: Best Play for Imperfect Players and Game Tree Search, submitted for publication 1993
D.A. McAllester: Conspiracy numbers for min max search, Artificial Intelligence 35 (1988) 287-310
A.J. Palay: Searching with probabilities, Pitman 1985
J. Pearl: Heuristics, Addison-Wesley 1984
R.L. Rivest: Game tree searching by min max approximation, Artificial Intelligence 34 (1988) 77-96
S. Russell and E. Wefald: Do the Right Thing, MIT Press 1991 (see especially chapter 4)
C.E. Shannon: Programming a computer for playing chess, Philos. Magazine 41,7 (1950) 256-275