Minimax Pathology and Real-Number Minimax Model

Mitja Luštrek
Jožef Stefan Institute, Department of Intelligent Systems
Jamova 39, 1000 Ljubljana, Slovenia
mitja.lustrek@ijs.si

Game-playing programs usually evaluate moves by searching the game tree using the minimax principle. Practice shows that deeper searches produce better results. However, theoretical analyses indicate that under apparently reasonable conditions, the behavior of minimax is pathological, i.e. deeper searches produce worse results. In this paper, a minimax model that – unlike most work in this area – uses real numbers for positions' values is introduced. It usually does not exhibit pathology. Mechanism that improves evaluation with depth is explained. Comparison to chess is made, showing the model not to be unrealistic.

1. Introduction

Game playing has long been a test bed for artificial intelligence research and many techniques developed there have found application elsewhere. Game-playing programs typically choose their moves by searching the game tree: they build the tree of possible future moves and positions to some arbitrary depth, heuristically evaluate the leaves, and then propagate their values to the root using the minimax principle. Practical aspects of the technique are well understood and it is known that everything else being equal, the deeper the tree is searched, the better the program plays. However, mathematical analyses indicate that under seemingly sensible conditions, minimaxing amplifies the error in the heuristic evaluation, so increased depth of search increases the error. This phenomenon was first observed by Beal [1] and was later termed the minimax pathology. Several explanations have been proposed, but no definite conclusion has been reached to date.

One of the properties of game trees of real games that can explain the absence of pathology is the similarity between a position's descendents. The majority of past research used two-value minimax models, i.e. games could only be lost or won. We use real numbers for positions' values, which makes it possible to model this similarity in a more natural way and enables a direct comparison to a game-playing program, since this is the way game-playing programs evaluate positions. The mechanism that improves the quality of evaluation with increased depth of search is analyzed mathematically.

The paper is organized as follows. Section 2 is an introduction to the minimax pathology. Section 3 presents our model of minimax. Section 4 gives a mathematical explanation of minimax with real-number values. Section 5 compares our model to a chess program. Section 6 concludes the paper and points out some areas for further research.

2. Minimax Pathology

Game tree consists of nodes representing positions and edges representing moves. Game scores are assigned to the leaves. Ideally these would be the scores of terminal positions, but since they can be determined only if the whole tree is built, which is usually computationally infeasible, the scores are generally heuristic evaluations of non-terminal positions. The tree has interchanging max and min plies[1]: in the root it is your turn and you choose the descendant with the highest value (trying to maximize your score), in the next ply it is your opponent's turn and he chooses descendants with the lowest values (trying to minimize your score), etc. Each internal node is assigned a backed-up value, which is the value of the chosen descendant. This is illustrated in Figure 1; thick edges mark the moves players choose.

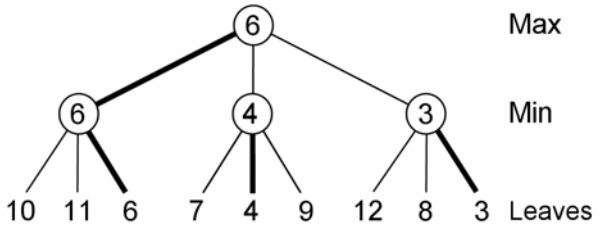

Figure 1. Game tree and minimax

In this section, two-value minimax model is analyzed: losses are marked with "–" and wins with "+". Negamax notation is used, i.e. nodes are marked as lost or won from the perspective of the player to move.

[1] Ply is a term for tree level used in game playing, i.e. one player's move.

Figure 2 shows two-value negamax representation of the tree in Figure 1.

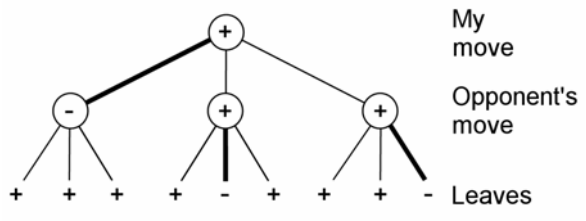

Figure 2. Two-value game tree and negamax

Game trees are assumed to have uniform branching factor b, depth of search d, and probability of a loss in i-th ply k_i. Plies are numbered upwards. The value of each leaf is independent of other leaves' values.

Two situations shown in Figure 3 are to be considered: a node has at least one lost descendant, in which case it is won because one can always choose move leading to the descendant lost for the opponent; or a node has only won descendants, in which case it is lost because all moves lead to positions won for the opponent.

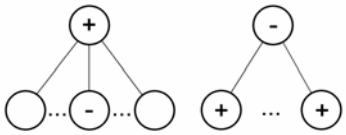

Figure 3. Types of nodes

Equation (1), which governs the relation between the values of k in consecutive plies, can be derived from the second situation (Figure 3 on the right).

$$k_{i+1} = (1 - k_i)^b \tag{1}$$

The goal is to calculate the probability of incorrectly evaluating the root given the probability of incorrect evaluation of the leaves. Two types of evaluation errors are possible: a loss can be mistaken for a win (false win) or a win for a loss (false loss). Let p_i and q_i be the probabilities of the respective types of errors in i-th ply. False wins occur in nodes where all descendants should be won (Figure 3 on the right), but at least one of them is a false loss. Equation (2) shows that p_{i+1} is the product of the probability of all descendants of a node in ply $i + 1$ being wins and the probability of at least one of them being a false win, divided by the probability of a node in ply $i + 1$ being a loss.

$$p_{i+1} = \frac{(1-k_i)^b(1-(1-q_i)^b)}{k_{i+1}} = 1-(1-q_i)^b \tag{2}$$

False losses occur in nodes where some descendants should be lost (Figure 3 on the left), but all of them are false wins instead, while all won descendants retain their true values. Equation (3) shows that q_{i+1} is the probability of a false loss occurring in a node in ply $i + 1$, summed over the number of possible lost descendants, and divided by the probability of a node in ply $i + 1$ being a win. The probability of a false loss occurring in a node in ply $i + 1$ with j lost descendants is the product of the probability of the node having j lost descendants, the probability of all of them being false losses and the probability of all the won descendants not being false wins.

$$q_{i+1} = \frac{\sum_{j=1}^{b} \binom{b}{j} k_i^j (1-k_i)^{b-j} p_i^j (1-q_i)^{b-j}}{1-k_{i+1}} \tag{3}$$

If only games where both sides have an equal chance of winning are considered, k_d must be 0.5 and equation (1) can be used to calculate k for other plies. After choosing p_0 and q_0, equations (2) and (3) can be used to calculate p_d and q_d. Overall error in the root can be defined as *position error* $p_d k_d + q_d (1 - k_d)$ or as *move error*, which is the probability of a wrong move to be chosen in the root due to position error in the root's descendants. If only cases where the moves lead to positions with different values are considered and b = 2, move error equals ½ ($p_d + q_d$). Either way, it turns out that with increasing d, the error converges towards 0.5, rendering minimax useless.

First attempts to explain the minimax pathology were made by Bratko and Gams [3] and Beal [2]. Both came to the conclusion that the reason minimax is effective in real games is that sibling nodes have similar values. Nau [5][6] used a simple game to show that strong dependence between a node and its descendants eliminates the pathology. Pearl [7] claimed that real games do not have such a strong dependence. He argued that early terminations (or blunders), which carry reliable evaluations, are the culprit. These attempts all used two-value model, while Sadikov et al. [8] used multiple values in their analysis of king and rook versus king chess endgame. They managed to explain the pathology, but it is not known how general their conclusion is.

Most of these explanations only show which property eliminates the pathology, while the exact mechanism is not sufficiently clear. Also, they are at best verified on special cases of real games.

3. Real-Value Model

In our model game tree is built from the root down. Static value 0 is assigned to the root and values of its descendants are distributed around it. The rationale for such a distribution is that descendants of a position are only one move away from it and are therefore unlikely to have a significantly different value. The process is repeated recursively on each descendant until the chosen depth is reached. Error is introduced as normally distributed noise in the leaves. Minimaxing is

performed on the original and the erroneous values. Unlike in two-value model, position error is defined as the difference between the correct and the corrupted backed-up value of a node.

The model has a number of parameters: branching factor b, type of distribution of nodes' static values around the static value of their parent, its standard deviation σ_v, the interval within which nodes' values must lie $[-m, m]$, and standard deviation of error σ_e. Only relative values of σ_v, m, and σ_e are important, so σ_v can be fixed to 1. In Figure 4 results for normal distribution of nodes' values, $m = \infty$, and $\sigma_e = 0.2$ are shown. The parameters are chosen rather arbitrarily, but their effect is examined later on. The results are averaged over 10,000 game trees for $b = 2$ and 1,000 game trees for $b = 5$, each time with 10 different error placements per tree. Only move error is plotted; position error behaves similarly.

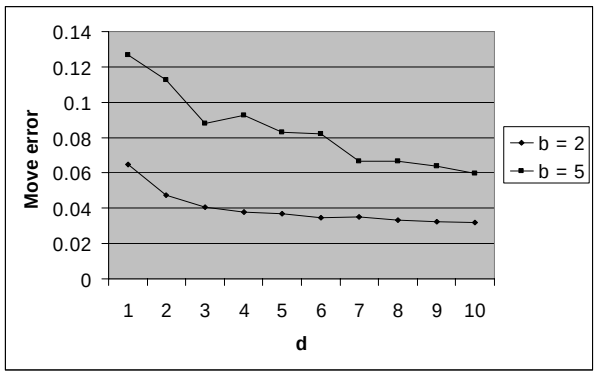

Figure 4. Move error in the root as a function of d

Normal, triangular, and uniform distributions of nodes values were tested with very similar results. Several values of m were tried. With $m < 3$ pathology was present, because nodes' values could not be properly distributed around the value of their parent. There was hardly any difference between $m \geq 4$ and $m = \infty$. Varying σ_e had little effect on the pathology. We can conclude that in our model, minimax is not pathological for nearly all parameter settings.

4. Mathematical Analysis

For the purpose of mathematical analysis, constant difference between the values of sibling nodes is assumed and only $b = 2$ is considered.

A node in ply $i + 1$ has two descendants with true backed-up values $\mu_{L\,i}$ (lower value) and $\mu_{i\,H}$ (higher value). Erroneous backed-up values are random variables L and H. In the leaves they are distributed normally with means $\mu_{0\,L}$ and $\mu_{0\,H}$ and standard deviation σ_e. Their probability density functions are given in equations (4).

$$f_{0L}(x) = \frac{1}{\sigma_e\sqrt{2\pi}} e^{\frac{-(x-\mu_{0L})^2}{2\sigma_e^2}} \qquad f_{0H}(x) = \frac{1}{\sigma_e\sqrt{2\pi}} e^{\frac{-(x-\mu_{0H})^2}{2\sigma_e^2}} \tag{4}$$

In a max ply $i + 1$, the value of a node is random variable $MAX_{i+1} = \max(L_i, H_i)$ with probability density function calculated according to equation (5).

$$f_{i+1MAX}(x) = f_{iH}(x)P(L < x) + f_{iL}(x)P(H < x) =$$
$$= f_{iH}(x)\int_{-\infty}^{x} f_{iL}(l)dl + f_{iL}(x)\int_{-\infty}^{x} f_{iH}(h)dh \tag{5}$$

Probability density functions of L_0, H_0, and MAX_1 are shown in Figure 5.

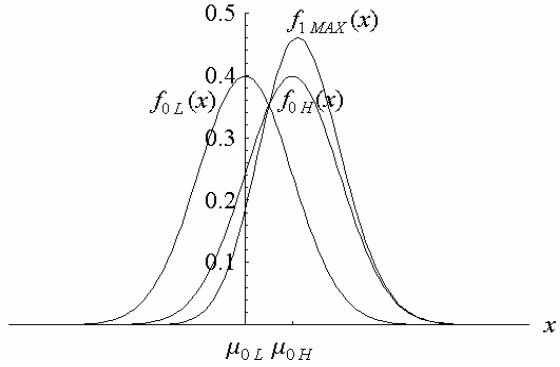

Figure 5. Probability density functions of the values of two sibling nodes in ply 0 and their parent

As can be seen in Figure 5, the curve of the parent is narrower than the curves of its descendents, meaning that position error of the parent is smaller.

Analogously to equation (5), the probability density function in a min ply is calculated according to equation (6).

$$f_{i+1MIN}(x) = f_{iL}(x)\int_{x}^{\infty} f_{iH}(h)dh + f_{iH}(x)\int_{x}^{\infty} f_{iL}(l)dl \tag{6}$$

Since probability density functions in ply 0 are given by equations (4), probability density function in any ply can be calculated by repeatedly using equations (5) and (6). This is shown in Figure 6; $\mu_{i\,H} - \mu_{i\,L} = 1$ for all i.

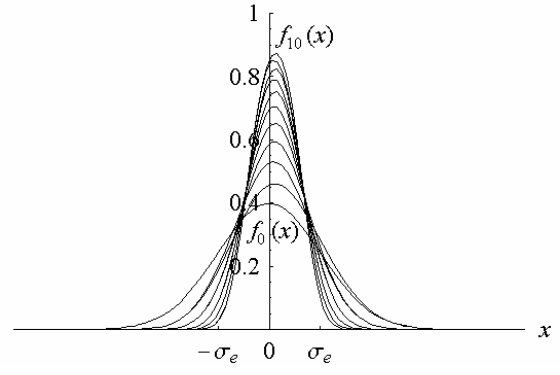

Figure 6. Probability density functions in plies 0–10

As can be seen in Figure 6, the higher a ply, the narrower the curve of the probability density function of a node's backed-up value. This means that position error decreases with depth of search.

Let ME_{i+1} be move error in ply $i + 1$. An erroneous move in ply $i + 1$ is chosen when the values of a pair of sibling nodes in ply i are switched, i.e. $L_i > H_i$, so ME_{i+1} is calculated according to equation (7).

$$ME_{i+1} = P(L_i > H_i) = \int_{-\infty}^{\infty} f_{iH}(h)\left(\int_{h}^{\infty} f_{iL}(l)dl\right)dh \qquad (7)$$

How move error changes with increasing depth of search is shown in Figure 7; ME_d is move error in the root when the depth of search is d, $\mu_{iH} - \mu_{iL} = 1$ for all i, and $\sigma_e = 1$.

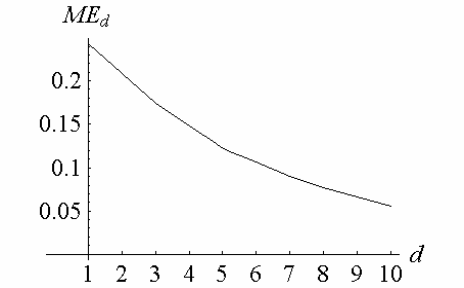

Figure 7. Move error in the root as a function d

5. Verification in Chess

Whether nodes' static values are indeed distributed around the static value of their parent was verified with the chess program Crafty [4]. For each of 450,000 game tree nodes visited in the course of a game, the differences between the static value of the node and the static values of all of its descendants were calculated. In Figure 8, the results are shown as the number of cases where the difference lies within an interval; the lower and upper 1% of the cases are omitted for clarity.

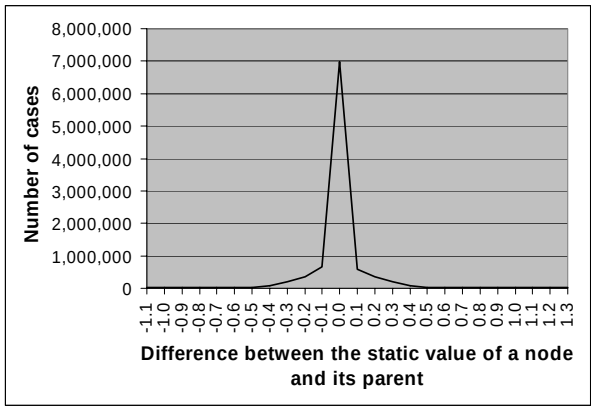

Figure 8. Distribution of the differences between the static value of a node and its parent

As can be seen in Figure 8, the distribution resembles normal, which is the default in our model. Since it was found that the type of distribution is not that important, we feel that the results from Crafty can be considered a confirmation of the model.

6. Conclusion

We designed a real-value minimax model with nodes' static values distributed around the static value of their parent. It was tested with a wide variety of parameter settings and only in the special case when the basic assumption of the model could not be properly expressed did it behave pathologically. The mechanism through which minimaxing reduces the noise introduced in the leaves of a game tree was explained in mathematical terms. The explanation is not as general as one might wish, but it appears to be essentially correct. It was shown that our model corresponds reasonably well to chess. From this we can conclude that one reason that makes minimax the algorithm of choice for game-playing programs is found and understood.

References

[1] D. F. Beal. An Analysis of Minimax. Advances in Computer Chess 2 (ed. M. R. B. Clarke), pp. 103-109, Edinburgh University Press, 1980

[2] D. F. Beal. Benefits of Minimax Search. Advances in Computer Chess 3 (ed. M. R. B. Clarke), pp. 17-24, Pergamon Press, 1982

[3] I. Bratko and M. Gams. Error Analysis of the Minimax Principle. Advances in Computer Chess 3 (ed. M. R. B. Clarke), pp. 1-15, Pergamon Press, 1982

[4] R. Hyatt. Home page. http://www.cis.uab.edu/info/faculty/hyatt/hyatt.html

[5] D. S. Nau. An Investigation of the Causes of Pathology in Games. Artificial Intelligence 19(3), pp. 257-278, 1982

[6] D. S. Nau. Pathology on Game Trees Revisited, and an Alternative to Minimaxing. Artificial Intelligence 21(1, 2), pp. 221-224, 1983

[7] J. Pearl. On the Nature of Pathology in Game Searching. Artificial Intelligence 20(4), pp. 427-453, 1983

[8] A. Sadikov, I. Bratko and I. Kononenko. Search vs Knowledge: Empirical Study of Minimax on KRK Endgame. Advances in Computer Games: Many Games, Many Challenges (eds. H. J. van den Herik, H. Iida, and E. Heinz), pp. 33-44, Kluwer Academic Publishers, 2003