Mach Learn (2006) 63:329–340
DOI 10.1007/s10994-006-6747-7

TECHNICAL NOTE

Learning long-term chess strategies from databases

Aleksander Sadikov · Ivan Bratko

Received: March 10, 2005 / Revised: December 14, 2005 / Accepted: December 21, 2005 / Published online: 28 March 2006
Springer Science + Business Media, LLC 2006

Abstract We propose an approach to the learning of long-term plans for playing chess endgames. We assume that a computer-generated database for an endgame is available, such as the king and rook vs. king, or king and queen vs. king and rook endgame. For each position in the endgame, the database gives the "value" of the position in terms of the minimum number of moves needed by the stronger side to win given that both sides play optimally. We propose a method for automatically dividing the endgame into stages characterised by different objectives of play. For each stage of such a game plan, a stage-specific evaluation function is induced, to be used by minimax search when playing the endgame. We aim at learning playing strategies that give good insight into the principles of playing specific endgames. Games played by these strategies should resemble human expert's play in achieving goals and subgoals reliably, but not necessarily as quickly as possible.

Keywords Machine learning · Computer chess · Long-term Strategy · Chess endgames · Chess databases

1. Introduction

The standard approach used in chess playing programs relies on the minimax principle, efficiently implemented by alpha-beta algorithm, plus a heuristic evaluation function. This approach has proved to work particularly well in situations in which short-term tactics are prevalent, when evaluation function can easily recognise within the search horizon the consequences of good or bad moves. The weakness of this approach may show in positions that require long-term planning beyond the search horizon.

Editors: Michael Bowling · Johannes Fürnkranz · Thore Graepel · Ron Musick

A. Sadikov (✉)· I. Bratko
University of Ljubljana, Faculty of Computer and Information Science, Tržaška 25,
1000 Ljubljana, Slovenia
e-mail: {aleksander.sadikov, ivan.bratko}@fri.uni-lj.si

Thus, a deficiency of minimax-based techniques for game playing is in the handling of long-term plans. Previous attempts at handling long-term plans include Advice Languages (Bratko, 1984) and chunking (George & Schaeffer, 1990).

Another aspect that is still deficient in computer game-playing is learning. Early attempts by Samuel (1967) in his checkers program were only partially successful. There have been some noticeable achievements: Tesauro's backgammon program (Tesauro, 1992), Buro's Othello player (Buro, 1999), and the chess playing program KNIGHTCAP (Baxter et al., 2000). However, successes of ML in games still seem to be relatively rare. The learning of long-term strategies is particularly weak and under represented. A thorough review of machine learning in game-playing is given in Fürnkranz (2001).

In this paper we investigate the learning of long-term strategies from computer-generated databases. Chess endgame tablebases, containing all possible positions for 5-piece and some 6-piece chess endgames, originally constructed by Thompson (1986, 1996), are readily available. Some of these endgames are very hard or impossible for the best human players to play reliably. The chess endgame databases contain a vast amount of detailed information, but this information is not in a form that a human player can use without computer. The problem is that raw information in these databases lacks conceptual structure. The databases lack explicit and economical concepts that would enable the human to understand, memorise and use the relevant knowledge. Our main goal in applying machine learning to chess databases is to help understand certain types of positions that are beyond human capabilities. So in this paper we attempt to convert detailed unstructured information into human-assimilable knowledge.

2. Method

Given a complete database for a chess endgame in which the stronger side can eventually win, our goal is to break down the endgame into stages that can be interpreted as a long-term strategy for playing this endgame. The main idea of our approach is based on the assumption that various stages are characterised by different sets of important features. In one stage, some things are important, when we enter another stage, other things become important.

2.1. Detecting borders between stages

The method that we propose here detects borders between (yet unknown) stages of play through the analysis of a set of position's attributes. We measure how relevant individual attributes are at different times, and then analyse the changes in attributes' importance during play. The assumption is that large changes indicate borders between stages of play. The details of our implementation of this idea are described in the following paragraphs.

We define the relevance of an attribute at a given point of play as the attribute's ability to measure progress of play the stronger side is making. This ability is evaluated by the attribute's strength to discriminate between positions of neighbouring levels in the database, i.e. distances from the final goal of play—for example the distance to mate in KRK (king and rook vs. king) or distance to winning the opponent's rook in KQKR (king and queen vs. king and rook). We chose to measure attributes' discrimination strength by Quinlan's information gain ratio (Quinlan, 1986).

Now assume we have a vector $\mathbf{A} = (a_1, a_2, \ldots)$ of position's attributes, and observe their information gain ratio for the following classification problems for $i = 1, 2, \ldots$: discriminate positions in the database of level i from those of level $i + 1$. All the positions are with the same side to move which means that positions at level $i + 1$ are one full move (i.e. two ply) apart from positions at level i. Information gain ratio of an attribute indicates to what extent each of the attributes on its own can recognise whether the position is of level i or $i + 1$. Let us denote the information gain ratio of attribute a_j with $g(i, j)$. Let $\mathbf{G}(i)$ stand for the vector of gain ratios of all the attributes with respect to this classification task, so $\mathbf{G}(i) = (g(i, 1), g(i, 2), \ldots)$. Now our assumption is that the behaviour of $\mathbf{G}(i)$ when level i changes is indicative of stages of play. If $\mathbf{G}(i)$ is similar to $\mathbf{G}(i + 1)$, this indicates that both levels belong to the same stage of play. Similar attributes are relevant to measuring the progress when moving from level $i + 1$ to level i. On the other hand, when the vectors $\mathbf{G}(i)$ and $\mathbf{G}(i + 1)$ are considerably different, this indicates that the important objectives of play differ between both levels. We chose to measure the similarity between gain vectors by correlation:

$$\mathrm{Corr}\,(i) = \frac{\mathrm{E}(\mathbf{G}(i) \cdot \mathbf{G}(i+1)) - \mathrm{E}(\mathbf{G}(i)) \cdot \mathrm{E}(\mathbf{G}(i+1))}{\sqrt{\mathrm{Var}(\mathbf{G}(i)) \cdot \mathrm{Var}(\mathbf{G}(i+1))}}. \tag{1}$$

Here, $\mathrm{E}(\mathbf{X})$ denotes the average value of the values in vector $\mathbf{X}$, and $\mathrm{Var}(\mathbf{X})$ the variance of values in $\mathbf{X}$.

The partitioning of the database into stages is thus carried out in the following steps:

1. Compute the gain vectors $\mathbf{G}(i)$ for all levels i.
2. For all levels i, compute correlations $\mathrm{Corr}(i)$ between the gain vectors of neighbouring levels i and $i + 1$.
3. Plot $\mathrm{Corr}(i)$ versus i. The values of i where $\mathrm{Corr}(i)$ has local minima are candidate points for borders between stages of play.

Once the database is segmented into subsets, that is stages of play, we can induce a classifier that classifies a given position into a corresponding stage of play. We are particularly interested in such classifiers that are comprehensible. The intention is that they characterise stages of play and thus give insight to a human into the long-term strategy for playing the endgame in question.

2.2. Playing the endgame

For each stage, we induce a regression function that approximates the distance-to-win for each position in that stage. Then we use this approximation of distance-to-win as a heuristic evaluation function for minimax search up to a chosen depth. The move played is decided by this minimax search. In the following paragraphs, we give some details of implementing these ideas in our experiments in this paper.

For each stage of play, we have a corresponding set of positions. These sets can be used as examples from which we can induce a classifier that discriminates between the stages. For a given position, such a classifier determines its stage of play. In our experiments we chose decision tree learning for this task.

For each stage of play we induce an evaluation function as an approximator of distance-to-win. In our experiments, we chose to use multivariate linear regression to approximate distance-to-win as a linear function of position's attributes. For each stage of play, we induce a linear regression function from the set of positions in the database that belong to that stage.

A position is evaluated as follows: first, the induced decision tree classifier is applied to determine the position's stage of play, and second, this stage's of play regression function is applied to the position to approximate its distance-to-win. It should be noted that the induced decision tree typically is not a perfect classifier, so it may classify some positions into incorrect stage of play, and consequently in such a case, a regression function is used that is intended for a different stage than the stage to which the position actually belongs. Therefore the so obtained distance-to-win predictor is typically imperfect for two reasons: first, the local linear regression functions make some numerical error, and second, the decision tree may select a wrong linear regression function.

We experimented with this approach in the simple KRK endgame, and the difficult KQKR endgame. We describe these experiments in the following sections. As a control experiment in the evaluation of stage partitioning, we compared the results so obtained with a "single stage" strategy. That is one without partitioning the database into stages, and inducing a global linear regression function for the whole database.

The tablebase for the KRK endgame used is available from the UCI Repository (Blake & Merz, 1998). It contains a subset (28,056 positions) of all legal positions, taking into account various symmetries. The tablebase for the KQKR endgame was constructed in the same fashion as Thompson's tablebases, however, only a random subset of about 60,000 positions was used for inducing a classifier that discriminates between stages of play. In both cases, the feature set used does not uniquely describe the positions. The classifiers were induced using the Orange (Demšar et al., 2004) tree learner (equivalent to C4.5 for all practical purposes), and due to our aim to obtain comprehensible classifiers we used forward pruning requiring that leaf nodes contain at least 1,000 examples. Classifiers induced without pruning were extremely complex.

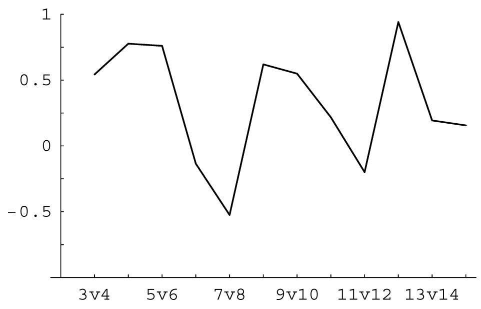
A simple mechanism is used to avoid cycling. If a repetition is about to occur, the program chooses the second-best move. Also, special code, acting similar to material element of the evaluation function, prevents both sides to lose material unnecessarily.

3. Experiments in king and rook versus king endgame

The attributes chosen for the learning about KRK were based upon sets of attributes used in some previous studies, e.g., Bratko (2001). Thus these attributes were not specially engineered for the application of our learning method in the KRK domain. On the other hand, some of the attributes were carefully defined in Bratko (2001) to support a particular winning method for KRK, so these attributes are rather domain specific.

Figure 1 shows the correlation plot measuring the similarity between adjacent gain vectors for the KRK endgame. The *x*-axis identifies the classification problem, that is distance-to-mate i versus $i + 1$, and the *y*-axis is the correlation value between this gain vector and the gain vector for the classification problem $i + 1$ versus $i + 2$. There are two distinct local minima, one separating levels 7 and 8, and the other separating levels 11 and 12. Accordingly

Fig. 1 Stage separation for the KRK endgame



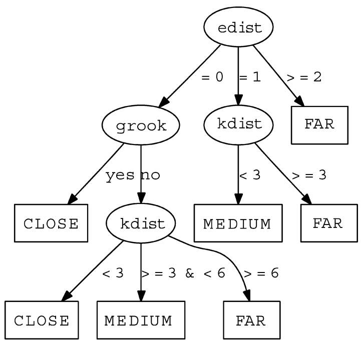


Fig. 2 Decision tree for classification into KRK stages. Attributes are: *edist* = distance in king moves of black king from nearest edge, *kdist* = distance between kings, *grook* = rook divides both kings with respect to the nearest edge to black king

we divided the KRK endgame into three stages: stage "CLOSE" comprised of levels 0 to 7, stage "MEDIUM" comprised of levels 8 through 11, and stage "FAR" comprised of levels 12 through 16. Levels 2 or less were omitted from the correlation plot because these levels contain a relatively low number of positions.

The induced decision tree for classification of positions into the three stages is shown in Fig. 2. It only uses three attributes, and is easy to interpret. On the basis of the differences between the neighbouring stages as given by the tree, we can extract the objectives of different stages of the game. The prevailing objective of stage FAR is to force the black king towards the edge. The objective of stage MEDIUM is to bring our king close to the opponent's king.

We experimented with several regression functions, using all twelve attributes. The regression function called GLOBAL was induced from all the positions in the database and applies to the whole domain. "Local" functions, called FAR, MEDIUM and CLOSE, were induced from the subsets of positions belonging to the corresponding stages of play. During play these local functions were used as described in Section 2. A position is classified by the tree of Fig. 2 into a stage, and the corresponding local regression function is applied. Note that this classifier is not perfect, so it may select a local function not intended for that position.

The distance-to-win predictors described above were used as evaluation function to play KRK with minimax search to depth 6 ply. The quality of play with various regression functions was assessed by two criteria: (a) *success rate*, the percentage of all positions in the endgame that the program can actually win, and (b) *suboptimality level*, how many more moves above the minimal number the player actually needs on average to win. The GLOBAL regression function has 58.43% success rate and suboptimality level of 29.71. The three local functions combined with the tree classifier have 100% success rate and suboptimality 4.22. These results indicate the benefit of our automatic decomposition of KRK into stages. A closer inspection shows that the GLOBAL function was affected by some of the attributes unsuitable for linear regression, in particular the attribute "white king distance to corner". This attribute behaves non-monotonically with respect to distance-to-win. The decomposition successfully coped with such "regression-unfriendly" attributes. However, it turned out in another experiment that, surprisingly, any of the three local functions when applied globally also plays comparably to the composite predictor. This result may on the other hand be interpreted as that the benefit of decomposition into stages was not essential.

4. Experiments in king and queen versus king and rook endgame

The KQKR endgame is usually won for the queen's side, except for rare special cases. Unlike the KRK endgame, which is trivial for good human players, KQKR is very hard, and

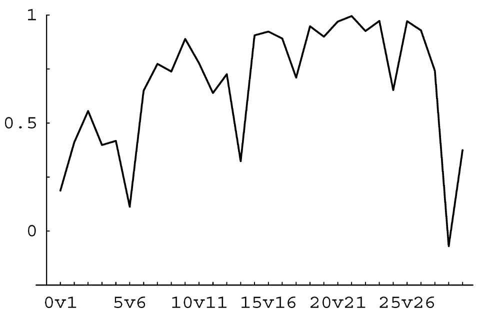


Fig. 3 Stage separation for the KQKR endgame

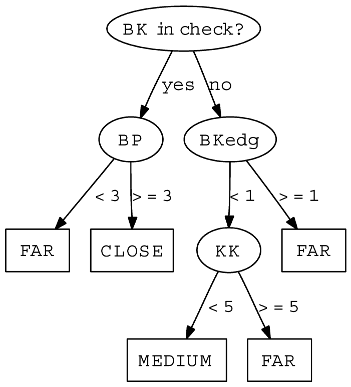
in practical play the stronger side may easily fail to win, as happened for example to Svidler, a leading grandmaster, in a game with grandmaster Gelfand.

We chose to use just simple attributes that typically make sense in such endgames and are not specific to KQKR. The following attributes are distances between any combination of two pieces: *wp* (distance between the two white pieces), *kk* (between two kings), *kr* (white king and rook), *qk* (queen and black king), *qr* (queen and rook), and *bp* (two black pieces). Attributes *wkedge*, *bkedge*, *wkcrn*, and *bkcrn* are the distances of white or black king to their nearest edge or corner, respectively. Finally, boolean attribute *bkchk* tells whether black king is in check. It should be noted that all the distances are measured in king moves. This is natural for distances involving kings, but debatable regarding other pieces.

Figure 3 shows the correlations between adjacent gain vectors for the KQKR endgame. There are three significant local minima: at move 6, at move 14, and at move 28. The reason why the minimum at move 24 was not considered for a split between stages, apart from the fact that one of the drops is higher in every other minima considered, is that this minimum still has a considerably high value of the correlation coefficient (over 0.65)—this signifies moderate to strong correlation and cannot be interpreted as a lack of correlation between corresponding vectors. The local minima we considered all have their corresponding correlation coefficients well below 0.4, signifying at most weak correlation.

In determining the stages of play, we ignore the rightmost minimum as it separates away only a relatively very small subset of positions at the far end. So on the basis of the first and the second minimum, we divided the KQKR endgame into three stages called CLOSE, MEDIUM and FAR, the borders between them being at move 6 and at move 14. These three stages contain 12.5%, 9.7%, and 77.8% of all the positions, respectively. We induced a decision tree for classification into these three stages shown in Fig. 4 (left). As the set FAR is far larger than both sets CLOSE and MEDIUM, we also tried to join the sets CLOSE and MEDIUM and induced a classifier for the resulting two-class problem (Fig. 4, right). Both trees in Fig. 4 are very similar, so in the interest of simplicity we decided to continue working with just two stages called CLOSE and FAR, where CLOSE now also includes the previous stage MEDIUM. The accuracy of the FAR vs. CLOSE decision tree is 81%. This is very low in the view of 77.8% majority of FAR.

Let us attempt an interpretation of the trees in Fig. 4. Can we learn something about how to play the KQKR endgame? Somewhat surprising, the attribute at the root is Black king in check. This rather short-term feature only becomes important in KQKR when the black pieces are not close together—in such cases Black king cannot defend the rook. This reflects the typical tactical idea in this endgame: White queen attacks both the Black king and rook at the same time, and when the king moves out of check the queen captures the rook. But the check is only effective when Black pieces are sufficiently far apart. This dependence is



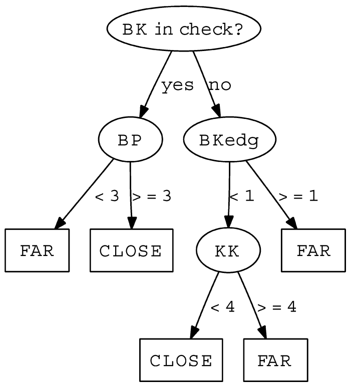


Fig. 4 On the left, decision tree for classification into three stages; on the right, tree for classification into "CLOSE" (i.e. CLOSE + MEDIUM) and FAR

reflected in the left hand branch of the tree where the Black king and rook are at least three king moves apart. It is precisely this distance when checking Black king becomes important, because at this distance or greater Black king cannot step next to the rook to defend it. Consequently, such positions are (probabilistically) classified as CLOSE. This branch in the tree implies a hint for White: force the Black pieces apart and then check the Black king. The right branch (Black king is not in check) says that for a position to be "close" to win, Black king must be on the edge and White king close to Black king (up to three king-moves). This contains another piece of advice for White. To make progress, Black king should be forced to an edge, and White king should approach the Black king. It should be noted that these suggestions for White correspond to the overall advice for KQKR in Dvoretsky's endgame manual (Dvoretsky, 2003). This is nice, but on the other hand this advice appears to be still too general for a human to play this endgame reliably and comfortably. In the experiments in computer play based on this stage decomposition of KQKR we found that this general knowledge requires rather deep minimax search to be effective. The amount of search required is beyond human's capability.

4.1. Playing the endgame

We induced three multivariate linear regression functions, using all eleven attributes, to predict distance-to-win for a given position: "GLOBAL" for the whole KQKR domain, "CLOSE" for the stage CLOSE, and "FAR" for the stage FAR. The numerical errors (difference between the predicted and the actual number of moves to win), measured by RMSE (root mean squared error), of these three functions, are: GLOBAL: 3.76, CLOSE: 1.97, FAR: 1.32. RMSE of the two local regression functions overall on the whole domain (CLOSE and FAR) is 1.46. This would be the error of the composite, piece-wise linear predictor comprising the two local linear predictors, if we had a perfect classifier for classifying positions into CLOSE vs. FAR. Of course, without the help of the database, we do not have such a perfect classifier. Instead, we used an approximate classifier, the right hand tree in Fig. 4. Due to high classification error of this tree, RMSE of this composite classifier is 3.73. Unfortunately, this is almost the same as the global linear regression function. Obviously, applying a local regression function on a position from a different stage causes numerical error to increase dramatically. This fact

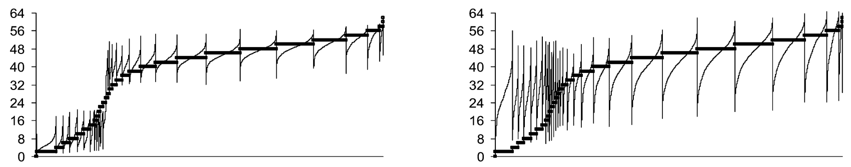


Fig. 5 Combined linear regression (left) and global linear regression (right)

indicates that an implementation of computer play based on the two-stage split CLOSE vs. FAR will have problems.

Figure 5 enables further analysis of the accuracy of prediction by these linear regression functions. The horizontal axis in these diagrams corresponds to the index of a KQKR position, so that according to this index the positions are ordered thus: first by their true distance-to-win, and second, the positions with the same true distance-to-win are ordered according to the predicted distance-to-win. The vertical axis represents the predicted distance-to-win. The diagrams in Fig. 5 show two huge problems of our prediction functions. First, there is a large overlap of predicted values between adjacent true distance-to-win levels. This means that using this as an evaluation function with minimax will have difficulties in determining the most promising successor position. Obviously, our evaluation functions poorly discriminate between positions whose true distances-to-win are similar. The second problem of the composite predictor with tree classifier, is the discontinuity at the border between the stages of play. Close to the borders, the positions from different stages are hard to compare because the local linear regression functions have different biases.

We experimented with these distance-to-win predictors as evaluation functions with minimax search. In the experiments, we varied the following: (1) predictor (GLOBAL, CLOSE applied globally on whole domain, FAR applied globally, composite CLOSE+FAR with the right tree of Fig. 4, and composite with the perfect classifier), and (2) depth of minimax search measured in plies. As in KRK, we observed the quality of play in terms of success rate and degree of suboptimality. Of course, the composite predictor with perfect classifier is of no practical significance (as it requires the database itself), but it is useful to assess the upper limits of performance of piece-wise linear predictors.

Table 1 gives the success rate and degree of suboptimality (in parentheses) for each combination of predictor and depth of search. The results show that this approach requires

Table 1 Performance depending on evaluation function used and depth of minimax

Evaluation function	Depth 8	Depth 10	Depth 12	Depth 14	Depth 16
Global LR	38.43% (+21.29)	93.85% (+11.04)	82.63% (+10.65)	99.16% (+4.65)	99.95% (+3.56)
Local LR with perfect classifier	72.08% (+16.54)	96.21% (+9.94)	97.92% (+5.92)	98.22% (+4.47)	98.56% (+2.91)
Local LR with tree classifier	17.28% (+24.23)	19.04% (+23.54)	21.77% (+22.68)	27.74% (+21.44)	44.32% (+17.80)
CLOSE LR used globally	14.12% (+25.44)	15.40% (+24.87)	16.83% (+24.28)	18.03% (+23.80)	19.86% (+23.09)
FAR LR used globally	15.99% (+24.72)	28.84% (+22.32)	43.70% (+20.20)	89.88% (+10.75)	98.46% (+5.26)

search of roughly at least 10 ply before it starts becoming successful. Searching to 14 or 16 ply attains close to perfect success rate, whereby needing on average some 3 to 5 moves above optimal. This performance is achieved by the global linear predictor, and similarly by the piece-wise-linear predictor with perfect classifier. Unfortunately, the performance of the composite predictor with imperfect classifier is significantly inferior. This result shows that in this case the decomposition of the endgame into two stages of play did not contribute to the playing performance.

5. Related work

Several researchers attempted to compress chess databases, and thus make them comprehensible to humans, by extracting from databases rules to be used by humans. Quinlan (1979, 1983) did early experiments in King-Rook-King-Knight (KRKN) endgame with his ID3 program and induced rules for classifying lost-in-2-ply and lost-in-3-ply positions. He found that hand-crafting relevant attributes was getting harder with increasing number of plies. For lost-in-4-ply, he tried to automatically generate relevant attributes, but there was no further report on that.

Shapiro and Niblett (1982) used structured induction for King-Pawn-King (KPK) endgame to alleviate the problem of acquiring good attributes. They constructed a hierarchical model consisting of two layers of decision trees. The tree in the upper layer divided the endgame into separate subproblems (e.g., KPK with the rook pawn), and the trees in the lower layer classified the positions as either won or drawn. Trees were induced with the ID3 program.

Bain and Srinivasan (1995) tried to learn a perfect player for the King-Rook-King (KRK) endgame using inductive logic programming methods. They managed to learn exact classifiers for "won in 0 moves" up to "won in 5 moves" positions. However, their approach resulted in a vast number of rules, making it hard to understand by humans.

There were also attempts to learn suboptimal playing strategies from endgame databases ("suboptimal" in the sense of winning in a number of moves larger than necessary). Human players rarely use optimal strategies, therefore this relaxation is very sensible. Morales (1994) developed a system called PAL, similar to Shapiro and Niblett's structured induction, but using inductive logic programming techniques. PAL learned to play the KRK endgame reasonably well. Muggleton's program DUCE (Muggleton, 1989) used constructive induction starting with primitive board features and occasionaly queried a domain expert. It learned to play part of the difficult King-Bishop-Bishop-King-Knight (KBBKN) endgame.

6. Discussion

Let us assess the results of our approach to the learning from databases. On the positive side, the results of learning in the KQKR endgame indicate some success with the proposed approach, given the difficulty of play in this endgame. KQKR is much more difficult than any other domain used in previous attempts at learning from chess databases. But let us analyse the achievements more systematically against the set goals of this work. The goal of learning was twofold: to extract descriptions that would (1) help the human player understand and play the endgame, and (2) would enable good computer play.

According to our experimental results, these goals were partially achieved. The decomposition into stages of play does give a human player some insight into the endgame and some hints about how to play the endgame. These insights are correct and useful, and are basically in line with general recommendations from chess endgame textbooks. However, at least for KQKR, the more difficult of the two experimental endgames, the extracted insights are rather general and are hardly sufficient to enable a human to win this endgame easily and reliably. Thus extracting more specific and powerful patterns remains the task for future work.

The induced knowledge, which also included the (piece-wise) linear regression distance-to-win estimate, was, in combination with minimax, converted into a playing algorithm. This enabled reliable play given a sufficient depth of search. On the critical side, however, the success of our approach regarding playing strength depends on how deep is sufficiently deep? Correct play at unlimited depth of search (or 60 ply in KQKR) is trivial and does not require any knowledge apart from the rules of the game. In the case of KQKR, it turned out that sufficient depth was 14 ply. This is incomparably better than playing without any knowledge, but still it appears a bit high and remains as an obvious challenge for future work. In fact, looking at the regression based prediction of distance-to-win in Fig. 5, reliable play emerging from these predictors may be a major positive surprise.

Our learned programs play reasonably well, but we were also interested in the style of play. Is it more human-like than the play resulting from the optimal database? Looking at games played by our program, they clearly follow the game decomposition plan into stages, achieving the next stage as a next subgoal. There are occasional slippages when immediately after attaining the next stage, the program drops back for a short time into the previous stage and then firmly re-attains the next stage again. The play is suboptimal in terms of the number of moves needed to win. So these games appear to could have been quite naturally played by a human player who is aware of the stage decomposition plan. However, it is hard to prove that these games are indeed more human-like than, say, the same endgames played by a chess program like FRITZ. We analysed games played by our program and those by FRITZ, and found that the differences in style are rather subtle. So it seems that a substantial psychological study would be needed to decide convincingly that our programs' play is a better model of human play.

In both KRK and KQKR, we compared experimentally the program's playing strength when using global regression and the more sophisticated regression that relies on the decomposition into stages of play. Decomposition into stages has not proved to be particularly effective in our minimax-based playing algorithm. This experimental observation is rather surprising. Counterintuitively, an evaluation function with better prediction accuracy, when used with minimax, performed inferior to a numerically less accurate predictor. In particular, the global linear regression function, or even a local linear regression function applied globally, played superior to more sophisticated, combined piece-wise linear predictors. How can this be explained? Probably this is due to discontinuities between stages of play (cf. regression functions in Fig. 5). The linear regression functions are biased toward average, so they have different biases for different stages of play. These functions are glued together by the decision tree classifier, which results in large discontinuities at the borders between the stages. This indicates that this way of exploiting the decomposition into stages has an unfortunate drawback. The fact that the tree classifier is imperfect is responsible for higher numerical prediction error of the composite predictor, but this seems not to be the critical deficiency of the composite predictor. The composite predictor performs inferior even when used with a perfect classifier. Therefore the difficulty regarding playing performance seems to come mainly from the discontinuities rather than from numerical inaccuracy.

Another problem with our decomposition, in which stages are defined in terms of distance-to-win, is that this distance may not reflect the "true" stages perfectly. In particular, distance-to-win may not necessarily correspond to the current objectives of play.

In KQKR, the playing algorithm was observed to be relatively more successful in the FAR stage than in the CLOSE stage. Why is FAR easier to play than CLOSE? The answer may be that FAR is dominated by clear goal-oriented behaviour (centralise king, force opponent's king towards edge), whereas CLOSE is dominated by precise tactics that require rather deep search, but not by some clear long-term goals that can be expressed in terms of simple measures, such as piece-to-piece distance or piece-to-corner distance. One interesting observation related to this is that our approach might nicely complement the standard approach to computer chess playing. The standard approach is particularly successful in short-term, tactics dominated play, and not in long-term, strategic play. This is just the opposite to our algorithm. So the obvious combination would be: rely on long-term strategic knowledge induced by our approach in the FAR stage, and then rely on the tactics-effective search of the standard approach in the CLOSE stage.

Let us mention some other limitations of our work. One obvious limitation is in the selection of attributes used in learning. There was no construction of new, stage-specific attributes. One would imagine that in the CLOSE stage of the KQKR endgame which seems to lack simple measures of progress, it should be possible to help the relatively deep tactical search by tactical patterns. Our induction algorithm did not reveal any such tactical patterns, possibly due to lack of relevant attributes among the chosen set of attributes. They should perhaps also include distances in terms of other piece moves—not only king-move distances. Then possibly endgame specific patterns could be automatically constructed. An example of such pattern-based attribute for KQKR is suggested by Dvoretsky (2003): control by the queen the square from which black rook could check white king after the king will have moved closer to the black king.

Appendix: Sample games

We have provided some commented games of our learned computer players on the web. They can be accessed at the following address: `http://www.ailab.si/sasha/MLJ/MLJgames.html`

References

Bain, M. & Srinivasan, A. (1995). Inductive logic programming with large-scale unstructured data. In K. Furukawa, D. Michie, and S. Muggleton (Eds.), *Machine Intelligence 14*. Oxford: Clarendon Press.

Baxter, J., Tridgell, A., & Weaver, L. (2000). Learning to play chess using temporal differences. *Machine Learning, 40*(3), 243–263.

Blake, C. L. & Merz, C. J. (1998). UCI repository of machine learning databases. `http://www.ics.uci.edu/~mlearn/MLRepository.html`, Department of Information and Computer Science, University of California, Irvine, CA.

Bratko, I. (1984). Advice and planning in chess endgames. In A. Elithorn and R. Banerji (Eds.), *Artificial and human thinking* (pp. 119–130). Amsterdam: North-Holland.

Bratko, I. (2001). *Prolog programming for artificial intelligence*, 3rd edn. Addison-Wesley Publishing Company.

Buro, M. (1999). How machines have learned to play Othello. *IEEE Intelligent Systems Journal, 14*(6), 12–14.

Demšar, J., Zupan, B., & Leban, G. (2004). Orange: From experimental machine learning to interactive data mining. `http://www.ailab.si/orange` (White Paper), Faculty of Computer and Information Science, University of Ljubljana.

Dvoretsky, M. (2003). *Dvoretsky's endgame manual*. Milford, CT: Russell Enterprises.

Fürnkranz, J. (2001). Machine learning in games: A survey. In J. Fürnkranz and M. Kubat (Eds.), *Machines that learn to play games*. New York, NJ: Nova Scientific Publishers.

George, M. & Schaeffer, J. (1990). Chunking for experience. *International Computer Chess Association Journal, 13*(3), 123–132.

Morales, E. (1994). Learning patterns for playing strategies. *International Computer Chess Association Journal, 17*(1), 15–26.

Muggleton, S. (1989). DUCE, an oracle based approach to constructive induction. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*. Morgan Kaufmann (pp. 287–292).

Quinlan, J. R. (1979). Discovering rules by induction from large collections of examples. *Expert Systems in the Micro Electronic Age*.

Quinlan, J. R. (1983). Learning efficient classification procedures and their application to chess endgames. *Machine Learning: An Artificial Intelligence Approach*, 463–482.

Quinlan, J. R. (1986). Induction of decision trees. In J. W. Shavlik and T. G. Dietterich (eds.), *Readings in machine learning*. Morgan Kaufmann, Originally published in *Machine Learning, 1*(1), 81–106.

Samuel, A. L. (1967). Some studies in machine learning using the game of checkers II—recent progress. *IBM Journal of Research and Development, 11*(6), 601–617.

Shapiro, A. D. & Niblett, T. (1982) Automatic induction of classification rules for a chess endgame. In M. R. B. Clarke (Ed.), *Advances in computer chess 3* (pp. 73–92). Oxford: Pergamon Press.

Tesauro, G. (1992). Practical issues in temporal difference learning. *Machine Learning*, *8*(3–4), 257–278.

Thompson K. (1986). Retrograde analysis of certain endgames. *International Computer Chess Association Journal, 9*(3), 131–139.

Thompson, K. (1996). 6-piece endgames. *International Computer Chess Association Journal, 19*(4), 215–226.