Available online at www.sciencedirect.com
SCIENCE DIRECT®
Theoretical Computer Science 349 (2005) 168–183
Theoretical Computer Science
www.elsevier.com/locate/tcs

Learning to score final positions in the game of Go

Erik C.D. van der Werf*, H. Jaap van den Herik, Jos W.H.M. Uiterwijk

Department of Computer Science, Institute for Knowledge and Agent Technology, Universiteit Maastricht, P.O. Box 616, 6200 MD Maastricht, The Netherlands

Abstract

This article investigates the application of machine-learning techniques for the task of scoring final positions in the game of Go. Neural network classifiers are trained to classify life and death from labelled 9 × 9 game records. The performance is compared to standard classifiers from statistical pattern recognition. A recursive framework for classification is used to improve performance iteratively. Using a maximum of four iterations our cascaded scoring architecture (CSA*) scores 98.9% of the positions correctly. Nearly all incorrectly scored positions are recognised (they can be corrected by a human operator). By providing reliable score information CSA* opens the large source of Go knowledge implicitly available in human game records for automatic extraction. It thus paves the way for a successful application of machine learning in Go.

Keywords: Go; Learning; Neural net; Scoring; Game records; Life and death

1. Introduction

Evaluating Go [1] positions is one of the hardest tasks in artificial intelligence (AI) [12,17]. In the last decades Go has received significant attention from AI research [2,14]. This was stimulated by Ing's million-dollar prize for the first computer program to defeat a professional Go player (it has expired unchallenged). Yet, despite all efforts, the best computer Go programs are still no match even for human amateurs of only moderate skill. Partially this is due to the complexity of Go, which makes brute-force search techniques infeasible on the 19 × 19 board. However, on the 9 × 9 board, which has a complexity between Chess and Othello [2], the *current* Go programs perform nearly as bad. The main reason lies in the lack of good positional evaluation functions. Many (if not all) of the current top programs rely on (huge) static knowledge bases derived from the programmers' Go skills and Go knowledge. As a consequence the top programs are extremely complex and difficult to improve. In principle a learning system should be able to overcome this problem.

In the past decade several researchers have used machine-learning techniques in Go. After Tesauro's [20] success story many researchers, including Dahl [4], Enzenberger [8] and Schraudolph et al. [18], have applied temporal difference (TD) learning for learning evaluation functions. Although TD-learning is a promising technique, which was underlined by NEUROGO's silver medal in 9 × 9 Go at the 8th Computer Olympiad in Graz [21], there has not been a major

* Corresponding author at: GN ReSound Research & Development, Algorithm R&D, Horsten 1, 5612 AX Eindhoven, Netherlands. Tel.: +31 40 2478336; fax: +31 40 2466712.
E-mail addresses: evdwerf@gnresound.com (E.C.D. van der Werf), herik@cs.unimaas.nl (H.J. van den Herik), uiterwijk@cs.unimaas.nl (J.W.H.M. Uiterwijk).

[1] For general information about the game including an introduction to the rules readers are advised to visit gobase.org [19].

0304-3975/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2005.09.045

breakthrough, such as in Backgammon, and we believe that this will remain unlikely to happen in the near future as long as most learning is done from self-play or against weak opponents.

Over centuries humans have acquired extensive knowledge of Go. Since the knowledge is implicitly available in the games of human experts, it should be possible to apply machine-learning techniques to extract that knowledge from game records. So far, game records have only been used successfully for move prediction [4,7,24]. However, we are convinced that much more can be learned from these game records.

One of the best sources of game records on the Internet is the no name Go server game archive [16]. NNGS is a free on-line Go club where people from all over the world can meet and play Go. All games played on NNGS since 1995 are available on-line. Although NNGS game records contain a wealth of information, the automated extraction of knowledge from these games is a non-trivial task at least for the following three reasons.

Missing information: Life-and-death status of blocks is not available. In scored games only a single numeric value representing the difference in points is available.

Unfinished games: Not all games are scored. Human games often end by one side resigning or abandoning the game without finishing it, which often leaves the status of large parts of the board unclear. [2]

Bad moves: During the game mistakes are made which are hard to detect. Since mistakes break the chain of optimal moves it can be misleading (and incorrect from a game-theoretical point of view) to relate positions before the mistake to the final outcome of the game.

The first step towards making the knowledge in the game records accessible is to obtain reliable scores at the end of the game. Reliable scores require correct classifications of life and death. This article focuses on determining life and death for final positions. By focusing on final positions we avoid the problem of unfinished games and bad moves during the game, which will have to be dealt with later.

It has been pointed out by Müller [13] that *proving* the score of final positions is a hard task. For a set of typical human final positions, Müller showed that extending Benson's techniques for proving life and death [1] with a more sophisticated static analysis and search, still leaves around 75% of the board points unproven. His program EXPLORER classified most blocks correctly, but did so by means of a heuristic classification. Moreover, it still left some regions unsettled (they should be played out further). Although this may be appropriate for computer–computer games, it can be annoying in human-computer games, especially under the Japanese rules which penalise playing more stones than necessary.

Since *proving* the score of most final positions is not (yet) an option, we focus on learning a heuristic classification. In this article our main focus is on learning the heuristic classification by a multi-layer perceptron (MLP). Consequently, the heuristic rules that are learnt for the classification will be implicitly contained in the numerical weights of the MLP, which are not easily understood by humans. However, it may be possible to use our framework to train alternative classifiers, such as a decision tree classifier, which could facilitate the extraction of heuristic rules that are more easily understood by humans. The next step is then to formulate the heuristics as lemmas, i.e., formulae which can be proven by using domain specific knowledge. As soon as a classification by lemmas is possible, scoring final positions is performed provably correct. This technique is adopted from the chess endgame two knights against pawn [5].

We believe that a learning algorithm for scoring final positions is important because: (1) it provides a more flexible framework than the traditional hand-coded static knowledge bases; and (2) it is a necessary first step towards learning to evaluate non-final positions. In general such an algorithm is good to have because: (1) large numbers of game records are hard to score manually; (2) publicly available programs still make too many mistakes scoring final positions; and (3) it can avoid unnecessarily long human-computer games.

The remainder of this article is organised as follows. Section 2 discusses the scoring method. Section 3 presents the learning task. Section 4 introduces the representation. Section 5 provides details about the dataset. Section 6 reports our experiments. Finally, Section 7 presents our conclusion and on-going work.

2. The scoring method

The two main scoring methods in Go are territory scoring and area scoring. Territory scoring, used by the Japanese rules, counts the surrounded territory plus the number of captured opponent stones. Area scoring, used by the Chinese

[2] In professional games that are not played on-line similar problems can occur when the final reinforcing moves are omitted because they are considered obvious.