Comparison of Bayesian Move Prediction Systems for Computer Go

Martin Wistuba, Lars Schaefers, and Marco Platzner

Abstract—**Since the early days of research on Computer Go, move prediction systems are an important building block for Go playing programs. Only recently, with the rise of Monte Carlo Tree Search (MCTS) algorithms, the strength of Computer Go programs increased immensely while move prediction remains to be an integral part of state of the art programs. In this paper we review three Bayesian move prediction systems that have been published in recent years and empirically compare them under equal conditions. Our experiments reveal that, given identical input data, the three systems can achieve almost identical prediction rates while differing substantially in their needs for computational and memory resources. From the analysis of our results, we are able to further improve the prediction rates for all three systems.**

I. Introduction

From the early days of research on Computer Go in the late sixties [1][2] until today, move prediction systems have been an integral part of strong Go programs. With the recent emergence of Monte Carlo Tree Search (MCTS) algorithms, the strength of Go programs increased dramatically to a level that appeared to be unreachable only seven years ago. MCTS is a simulation-based search method that learns a value function for game states by consecutive simulation of complete games of self-play using semi-randomized policies to select moves for either player. The design of these policies is key for the strength of MCTS Go programs and has been investigated by several authors in recent years [3][4][5][6][7]. Move prediction systems can serve as an important building block for the definition of MCTS policies as has been shown in, e.g., [5] and [8]. Several such systems have been developed especially for the use with Computer Go and prediction rates between 33.9% [4] and 34.9% [5] have been reported. However, these prediction rates have been produced under pretty unequal conditions using, e.g., different sizes of training data sets (ranging from 652 [5] to 181,000 [9] games) or different kinds and numbers of patterns (ranging from about 17,000 [5] to 12M [9]). Additionally, apart from the prediction rate, one might be interested in the technique's needs for computation and memory resources.

The contribution of this paper is the review of three prediction systems presented in the literature, namely those of Stern, Herbrich and Graepel [9], Weng and Lin [10] and Coulom [5], and their comparison under equal conditions for move prediction in the game of Go. Here, *equal* means that we use the same training and test data sets as well as the same set of shape and feature patterns to represent board configurations and move decisions.

All authors are with the University of Paderborn, Germany (email: mwistuba@mail.upb.de, {slars,platzner}@uni-paderborn.de)

The purpose of our work is the search for a time and memory efficient online algorithm for continuous adaptation of a move prediction system to actual observations which, as we believe, is a promising approach to further increase the strength of Go programs. In a larger context, the goal is to migrate knowledge gathered during MCTS search in certain subtrees into a prediction model that itself is used all over the search space to provide prior action value estimates and to guide playouts. This might help to reduce the horizon effect and to understand more complex positions.

Obviously, the two filtering methods will be amenable to continuous adaptation. Additionally, we included Coulom's method in our study as a reference point as the best prediction rates have been reported for this method and it is the most used approach in state of the art Go programs.

Section II gives a brief summary of related work on move prediction systems and provides background information about move prediction models and pattern recognition with a special focus on the game of Go. We present the algorithms used for the comparison in Section III. Section IV describes our experimental setup and the comparison results. Finally, Section V concludes our work and gives an outlook to future work.

II. Background and Related Work

This section gives a brief introduction to the game of Go and the abstraction of moves with patterns, and reviews methods for move ranking presented in the literature.

A. Game of Go

The game of Go is an ancient Asian two-player board game that is known to exist for more than 2,000 years. The board is made up of N horizontal and N vertical lines that build a grid with N^2 crossings. N is typically 19 but also 9 and 13 is commonly used. We have a black and a white player where each one is equipped with a large number of black resp. white pieces. The players alternate in placing one of their pieces on one of the empty crossings. Once a piece is placed on the board, it is never moved again. A player has the opportunity to pass, i.e. not to place a piece. At the start of the game, the board is empty and the black player makes the first move. If one player manages to surround a group of opponent stones, those opponent stones become prisoners and are removed from the board. The game ends after two consecutive pass moves. The goal for each player is to occupy or surround the largest part of the board with stones of their color.[1] Go games played by professionals are said to have an average length of about 150 moves with an average number of about 250 possible

[1]see http://gobase.org for more information

choices per move. The number of legal positions in 19x19 Go is about 2×10^{170} [11]. In more than 40 years of research no static evaluation function for intermediate game positions was found good enough to achieve strong performance with traditional depth limited alpha-beta search [12] or conspiracy number search [13]. The vast amount of positions paired with the absence of strong evaluation functions makes Go extremely challenging and made it a popular testbed for AI research in recent years. A great deal of information on current developments in Computer Go and the most often used MCTS algorithms can be found in [14] resp. [15].

B. Bayes Theorem

As we consider three *Bayesian* move prediction systems in this paper, the Bayes theorem takes a central role so we will briefly introduce it here. From the Bayesian viewpoint, probabilities can be used to express degrees of belief. Given a proposition A, a *prior* belief in A of $P(A)$ and a probability model $P(A|B)$ that expresses the *likelihood* of A observing supporting data B, Bayes Theorem relates the *posterior probability* $P(B|A)$ of observing data B given A in future experiments as follows:

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)}. \tag{1}$$

Hence, having a probability model $p(D|\boldsymbol{\theta})$ for a move decision D given some model parameters $\boldsymbol{\theta}$, one might infer the posterior probability of the model parameters $p(\boldsymbol{\theta}|D)$ observing some move decisions D using Bayes Theorem. Inference is a basic element of the prediction systems considered in this paper. A challenging point in this inferences is the possible emergence of difficult to handle integrals in the computation of $p(D)$ when considering continuous distributions: $p(D) = \int p(D|\boldsymbol{\theta})p(\boldsymbol{\theta})d\boldsymbol{\theta}$.

C. Terminology

We formalize a game as a tuple $G := (S, A, \Gamma, \delta)$, with S being the set of all possible states (i.e. game positions), A the set of actions (i.e. moves) that lead from one state to the next, a function $\Gamma : S \rightarrow \mathcal{P}(A)$ determining the subset of available actions at a state and the transition function $\delta : S \times A \rightarrow \{S, \emptyset\}$ specifying the follow-up state for a state-action pair where $\delta(s, a) = \emptyset$ iff $a \notin \Gamma(s)$.

In the remainder of this paper, state-action pairs (s, a) will be abstracted with a team of patterns with assigned strength values. We write the vector of pattern strength values as $\boldsymbol{\theta}$.

D. Modeling

In this paper, we consider parameterizable, distributive move prediction models $\mathcal{M}$ that, given a game state $s \in S$ and a parameter vector $\boldsymbol{\theta}$, yield a probability distribution over all available actions $a \in \Gamma(s)$, hence, we search for models $\mathcal{M} : S \times A \rightarrow P(a|s, \boldsymbol{\theta})$ with $\forall_{a \notin \Gamma(s)} : P(a|s) = 0$, $\forall_{a \in \Gamma(s)} : P(a|s) \geq 0$ and $\sum_{a \in \Gamma(s)} P(a|s \in S) = 1$.

A common technique for the modeling of games is the abstraction of the state-action space with patterns that represent certain move properties. Shape patterns that describe the local board configuration around possible moves and thus around yet empty intersections were already used in the early time of research on Go [16]. Stern et al. [9] describe diamond shaped pattern templates of different sizes around empty intersections to abstract the state-action space as shown in Fig. 1. In a separate harvesting step, they process the training data consisting of move decisions to build a database of frequently appearing shape patterns. Here, shape patterns are regarded to be invariant to rotation, translation and mirroring.

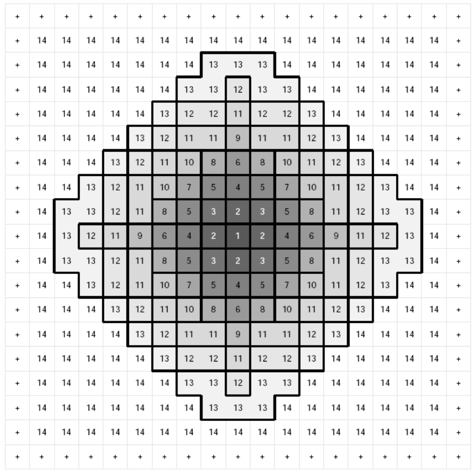


Fig. 1: Shape-Pattern templates of sizes 1 to 14 (Source: [9])

In addition to shape patterns, so called feature patterns, that represent non-shape properties of state-action pairs are used. A list of the patterns used in our experiments is given later on in Section IV.

Using the pattern system a move decision can be represented in form of a list of patterns (representing the possible moves) among one is marked as the final choice. Without loss of generality, we will assume that the final choice is always the first element in this list. A move decision can now be regarded as a competition between patterns that is won by the pattern representing the final move choice. By observing such competitions we can assign strength values θ_i to each of the patterns. Hence, a move decision yields a strength parameter vector $D = (\theta_1, \theta_2, \ldots, \theta_k)$ where θ_1 is the strength of the winning pattern.

Given the strength values θ_i and θ_j of two competitors, several probability models were developed to estimate the outcome of competitions between i and j. We will briefly describe two of them that are used by the prediction systems considered in this paper.

The **Thurstone-Mosteller Model** (TM) is used by Stern, Herbrich and Graepel [9] and models the players strength to be Gaussian distributed with $\mathcal{N}(\theta_i, \beta_i^2)$. Thus, in addition to the strength θ_i each player is also assigned a variance β_i^2 that models the uncertainty about the players actual performance. The probability that player i beats player j is than estimated by

$$P(i \text{ beats } j) = \mathbf{\Phi}\left(\frac{\theta_i - \theta_j}{\sqrt{\beta_i^2 + \beta_j^2}}\right),$$

where $\mathbf{\Phi}$ denotes the cumulative distribution function of the standard normal distribution.

Coulom [5] uses the **Bradley-Terry Model** (BT) [17] that is based on a logistic distribution to predict the outcome of games between two individuals with given strength values θ_i resp. θ_j. It estimates the probability that i beats j by

$$P(i \text{ beats } j) = \frac{e^{\theta_i}}{e^{\theta_i} + e^{\theta_j}}$$

The BT Model is also the base for the well known Elo rating system [18] that is used e.g. to rate human chess players.

Hunter [19] derived several generalizations of the BT model among one allows the prediction of the outcome of games between an arbitrary number of teams of individuals with the simplifying assumption that the strength of a team equals the sum of the strength of its members:

$$P(\text{1-2 beats 4-2 and 1-3-5}) = \frac{e^{\theta_1+\theta_2}}{e^{\theta_1+\theta_2} + e^{\theta_4+\theta_2} + e^{\theta_1+\theta_3+\theta_5}} \quad (2)$$

The use of this generalization to teams allowed Coulom to characterize each possible move at a given board position with a team of patterns.

Weng and Lin [10] support both prediction models, TM and BT. In this paper we concentrate on their BT based variant.

E. Ranking

We will now review three methods for the learning of model parameters from a given set of move decisions. As the main objective of this paper is the comparison of these approaches under equal conditions, we had to modify some of the algorithms in a straight forward manner to make them applicable to the same training data. The modifications are detailed in section III.

1) Bayesian Full Ranking: The probability model for game decisions in this approach developed by Stern et al. [9] is based on the TM model. In addition to the pattern strength that is modeled as a Gaussian, they account for varying performances or playing styles of players that produced the training set by adding a fixed variance β^2 to a pattern's strength distribution. Hence, the performance of a pattern i with strength $\theta_i = \mathcal{N}(\mu_i, \sigma_i^2)$ is given by $x_i = \mathcal{N}(\theta_i, \beta^2)$.

Then, given a board position with k possible moves and corresponding strength values $\theta_1, \ldots, \theta_k$, they give the following joint distribution for the probability that the pattern indexed with 1 is the best performing move:

$$p(i^* = 1, \boldsymbol{x}, \boldsymbol{\theta}) = \prod_{i=1}^{k} s_i(\theta_i) \prod_{j=1}^{k} g_j(x_j, \theta_j) \prod_{m=2}^{k} h_m(x_1, x_m), \quad (3)$$

where

$$s_i = \mathcal{N}(\theta_i; \mu_i, \sigma_i^2),$$
$$g_j = \mathcal{N}(x_j; \theta_j, \beta^2) \text{ and}$$
$$h_m = \mathbb{I}(x_1 > x_m).$$

Here, $\mathbb{I}(cond.)$ is the indicator function that equals 1 in case the given condition holds and 0 otherwise. The distribution is a product of small factors and can be represented as a graphical model, the so called factor graph. Fig. 3 in Section III shows an example of such a graph. For tree like factor graphs, there exist message passing algorithms that allow for efficient Bayesian inference in the graph by exploiting the graphical structure of the underlying distribution[2]. Except of the h_m, all factors yield Gaussian densities. In order to keep computations simple, emerging non-Gaussian distributions are approximated with Gaussians. The resulting inference algorithm is called Expectation Propagation [21].

As a result, we can infer new values for the pattern strength distribution $\boldsymbol{\theta}$ by incorporating observations from one move decision. This new values can now act as the prior for incorporating the next observation. This makes the method a filtering algorithm also called *assumed density filtering*.

A drawback of the initial model presented was the integration of binary non-shape move properties (so called features) in form of an additional feature vector into the shape-patterns. This entailed an exponential dependency of the number of features on the total number of patterns to evaluate, and accordingly limited the number of usable features.

2) Bayesian Approximation for Online Ranking: Weng and Lin [10] present a more general Bayesian approximation framework for online ranking of players from games with multiple teams and multiple players and compare their method to the online ranking system TrueSkill$^{\text{TM}}$ [22] that was developed and is used in commercial products by Microsoft. TrueSkill has a lot of similarities to the Bayesian Full Ranking Model described above. The presented framework can be used with the Bradley-Terry as well as the Thurstone-Mosteller and other ranking models. Like the Bayesian Full Ranking system of Stern et al., the players' strength values θ_i are updated after each new observation and are assumed to be normal distributed with mean μ_i and variance σ_i^2, i.e. $\theta_i = \mathcal{N}(\mu_i, \sigma_i^2)$. As a main achievement they were able to construct computationally light weight approximated Bayesian update formulas for the mean and variances of the θ_i by approximating the integrals that naturally show up in Bayesian inference analytically through an application of *Woodroofe-Stein's identity*.

In this paper we applied the algorithm based on the Bradley-Terry model to the game of Go in order to compare it to the approaches of Stern et al. and Coulom.

3) Minorization-Maximization: In contrast to the filtering algorithms of Stern et al. and Weng et al., Coulom proposed a whole-history rating concept that iterates several times over the training data with the objective to find the

[2]See [20] for more information on message passing in factor graphs.

pattern strength values that maximize the probability of the given move decisions from the training set according to the generalized Bradley-Terry model (Equation 2). Following the work of Hunter [19], Coulom iteratively applied the concept of Minorization-Maximization (MM) on

$$L = \prod_{j=1}^{N} P(D_j|\boldsymbol{\theta}), \tag{4}$$

where N is the number of move decisions in the training set and $P(D_j|\boldsymbol{\theta})$ is the generalized Bradley-Terry model. L can be written as a function of each of the single strength parameters $\gamma_i = e^{\theta_i}$. As illustrated in Fig. 2, MM is an optimization algorithm where, starting from an initial guess γ_0, a *minorizing* function $m(\cdot)$ for $L(\cdot)$ is build in γ_0 (i.e. $m(\gamma_0) = L(\gamma_0)$ and $\forall\gamma : m(\gamma) \leq L(\gamma)$) so that its maximum can be given in closed form. Finally the maximum γ_1 of m is choosen as an improvement over γ_0.

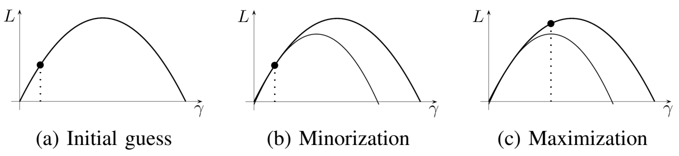


Fig. 2: Minorization-Maximization (Source: [5])

By using the generalized Bradley-Terry model, Coulom benefits from the ability to abstract each move with a group of patterns and thus, to use apart of shape patterns a larger number of additional feature patterns like e.g. the distance to the move played just before.

III. Algorithms

In this section, we give a more detailed presentation of the algorithms used in our experiments. These are not always the original algorithms as they were presented in the before mentioned literature, but might contain adaptations with the objective to support the core learning procedures of the different algorithms with the same amount and quality of data and thus, to create equal conditions for a head-to-head comparison. We give a detailed description in each case adaptations were made.

The input for all presented algorithms is identical: A number of game positions each annotated with the single move that is regarded to be the best. Once a new position is seen, each move is characterized with a team of patterns. Hence, after this preprocessing step, the input for all presented algorithms is a number of competitions between pattern teams among one team is marked as the winning team. For the remainder of this paper we define all variables related to the winning team to be indexed with 1.

Another common element of all considered methods is the definition of a ranking model, that results in a single function L that represents the likelihood of the observed data in the face of given pattern strength parameter values. Using Bayes-Theorem in the one or other way, this function is used to update strength parameters with the objective to maximize L.

A. Bayesian Full Ranking

The probability model used for inference in the Bayesian Full Ranking method of Stern et al. [9] is given by Equation 3. This distribution can be represented with a factor graph where each variable and each factor is represented by a node. The edges connect each factor node with the nodes of those variables the factor depends on. Stern et al. proposed a model that abstracts each move with exactly one shape-pattern with some binary non-shape properties encoded in it. We extended the model by introducing pattern teams as they were used in Coulom [5] by adding another level to the graph. This allows for the use of more non-shape patterns and makes a head-to-head comparison with Couloms approach possible. As some patterns can be member of more than one team, the graph might contain several cycles. Fig. 3 shows the factor graph of this model with the extension to pattern teams. In this figure, θ_i now represents a team of patterns instead of a single pattern and θ_{ij} a member of team i. In order to cope with the introduced cycles we used so called *loopy-belief propagation* for message passing. The possibility to do this was already mentioned by Stern himself in [23]. As the message passing algorithm is rather involved we cannot give the complete algorithm here but refer to [23] for a description of message passing in this particular setting with a derivation of all the update equations.

In the remainder of the paper we call the modified, pattern teams related approach *Loopy-Bayesian Ranking*.

B. Bayesian Approximation for Online Ranking

As for the Bayesian Full Ranking model, Weng and Lin use a probability model for the outcome of a competition between k teams that can be written as a product of factors that each involve only a small number of variables:

$$L = \prod_{i=1}^{k} \prod_{q=i+1}^{k} P(\text{outcome between team } i \text{ and team } q),$$

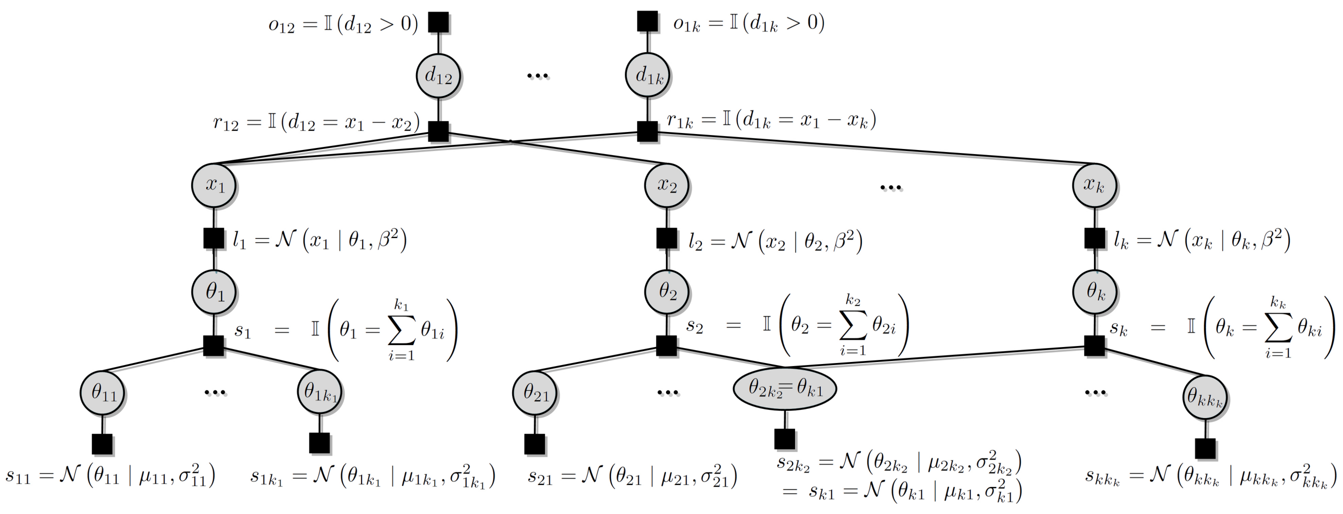
where P is given by the Bradley-Terry model.

For their system they consider a complete ranking of the teams whereas in case of a move decision in Go we have exactly one winning team and all the others ranked second. Hence, in addition to the aforementioned function L, we made experiments with a simplified function

$$L_{Go} = \prod_{i=2}^{k} P(\text{team 1 beats team } i).$$

Recall that in this paper the winning team has always index 1. In all our preliminary experiments we achieved significantly better prediction rates using L_{Go} instead of L[3]. Hence, in section IV all experiments were made using L_{Go}.

[3]In a recent discussion, Weng pointed out, that the use of L_{Go} might introduce some undesired bias in the parameter updates, as the variance of the winning team gets reduced much faster than for the losing teams.



$o_{12} = \mathbb{I}(d_{12} > 0)$
$o_{1k} = \mathbb{I}(d_{1k} > 0)$
d_{12}
d_{1k}
$r_{12} = \mathbb{I}(d_{12} = x_1 - x_2)$
$r_{1k} = \mathbb{I}(d_{1k} = x_1 - x_k)$
x_1
x_2
x_k
$l_1 = \mathcal{N}(x_1 \mid \theta_1, \beta^2)$
$l_2 = \mathcal{N}(x_2 \mid \theta_2, \beta^2)$
$l_k = \mathcal{N}(x_k \mid \theta_k, \beta^2)$
θ_1
θ_2
θ_k
$s_1 = \mathbb{I}\left(\theta_1 = \sum_{i=1}^{k_1} \theta_{1i}\right)$
$s_2 = \mathbb{I}\left(\theta_2 = \sum_{i=1}^{k_2} \theta_{2i}\right)$
$s_k = \mathbb{I}\left(\theta_k = \sum_{i=1}^{k_k} \theta_{ki}\right)$
θ_{11}
θ_{1k_1}
θ_{21}
$\theta_{2k_2} = \theta_{k1}$
θ_{kk_k}
$s_{11} = \mathcal{N}(\theta_{11} \mid \mu_{11}, \sigma_{11}^2)$
$s_{1k_1} = \mathcal{N}(\theta_{1k_1} \mid \mu_{1k_1}, \sigma_{1k_1}^2)$
$s_{21} = \mathcal{N}(\theta_{21} \mid \mu_{21}, \sigma_{21}^2)$
$s_{2k_2} = \mathcal{N}(\theta_{2k_2} \mid \mu_{2k_2}, \sigma_{2k_2}^2) = s_{k1} = \mathcal{N}(\theta_{k1} \mid \mu_{k1}, \sigma_{k1}^2)$
$s_{kk_k} = \mathcal{N}(\theta_{kk_k} \mid \mu_{kk_k}, \sigma_{kk_k}^2)$

Fig. 3: Factor graph of Bayesian Full Ranking Model (as seen in [23]) extended to pattern teams

The actual complete algorithm for the update of the pattern strength parameter vectors $\boldsymbol{\mu}, \boldsymbol{\sigma^2}$ on the observation of a new competition result towards maximizing L_{Go} is shown in Algorithm 1. These slightly modified algorithm is termed *Bayesian Approximation Ranking* in the remainder of the paper.

Algorithm 1: Bayesian Approximation Ranking

```
input : $(\theta_1, \sigma_1^2), \ldots, (\theta_k, \sigma_k^2)$, with $(\theta_1, \sigma_1^2)$ winning
output: Updated values $\boldsymbol{\mu}$ and $\boldsymbol{\sigma^2}$
$\beta^2 := 13$
for $i = 1, \ldots, k$ do
    $\mu_i = \sum_{j=1}^{n_i} \mu_{ij}$
    $\sigma_i^2 = \sum_{j=1}^{n_i} \sigma_{ij}^2$
end
                              /* Update team strength */
for $i = 1, \ldots, k$ do
    for $q = 1, \ldots, k, rank(i) \neq rank(q)$ do
        $c_{iq} = \sqrt{\sigma_i^2 + \sigma_q^2 + 2\beta^2}$
        $\hat{p}_{iq} = \frac{e^{\mu_i / c_{iq}}}{e^{\mu_i / c_{iq}} + e^{\mu_q / c_{iq}}}$
        if $i = 1$ then
            $\Omega_1 \leftarrow \Omega_1 + \sigma_1^2 \frac{\hat{p}_{q1}}{c_{1q}}$
            $\Delta_1 \leftarrow \Delta_1 + \frac{\sigma_1}{c_{1q}} \frac{\sigma_1^2}{c_{1q}^2} \hat{p}_{1q} \hat{p}_{q1}$
        end
    end
    if $i > 1$ then
        $\Omega_i \leftarrow -\sigma_i^2 \frac{\hat{p}_{i1}}{c_{i1}}$
        $\Delta_i \leftarrow \frac{\sigma_i}{c_{i1}} \frac{\sigma_i^2}{c_{i1}^2} \hat{p}_{i1} \hat{p}_{1i}$
    end
                              /* Update pattern strength */
    for $j = 1, \ldots, k_i$ do
        $\mu_{ij} \leftarrow \mu_{ij} + \frac{\sigma_{ij}^2}{\sigma_i^2} \Omega_i$
        $\sigma_{ij}^2 \leftarrow \sigma_{ij}^2 \cdot \max\left\{1 - \frac{\sigma_{ij}^2}{\sigma_i^2} \Delta_i, \epsilon\right\}$
    end
end
```

C. Minorization-Maximization

As described in II-E3, Coulom [5] uses the Minorization Maximization algorithm (MM) to obtain pattern strength parameter values that maximize function L as given in Equation 4. Given a number of N competitions, i.e. move decisions, MM requires us to iterate a number of times over the complete training set to update the strength values $\gamma_i = e^{\theta_i}$ of all patterns until convergence. Coulom derived the following update formula for the individual pattern strength parameters:

$$\gamma_i \leftarrow \frac{W_i}{\sum_{j=1}^{N} \frac{C_{ij}}{E_j}}, \tag{5}$$

where N is the number of competitions in the training set, W_i is the number of times pattern i was a member of the winning team in all N competitions, C_{ij} is the strength of the team-mates of pattern i in competition j and E_j is the sum of the strength of all teams competing in j. For more details on the derivation of the update formula 5 we refer to Coulom's paper [5]. We applied this approach following an implementation example published by Coulom on his homepage. Here, after a first update with Eqn. 5 for all gamma values, he selects the feature group that lead to the biggest improvement towards minimizing L during its last update also for the next update, until the last change of the logarithm of L was smaller than some threshold for each feature group. In this comparison we set the threshold to 0.001.

D. Probabilistic Ranking

In our comparison we also include the extremely simple and straight forward approach of just taking the relative winning rates of each pattern in the training set directly as their strength values. Thus, for each pattern i we have $\theta_i = W_i / A_i$, where W_i is again the number of competitions in which pattern i was a member of the winning team and A_i is the number of times pattern i was a member of any team. If a move is represented by a pattern team F, we define its strength by $\theta_F = \sum_{i \in F} \theta_i$.

IV. Experiments

We made a series of experiments with the four algorithms presented in section III. The main focus was on the comparison of the core learning routines of all approaches under equal conditions, with respect to the achievable prediction performance as well as the time needed to learn model parameters. For all approaches, the procedure to produce predictions, ones the model parameters were trained, is rather equal and simple, so we did not investigate computational requirements for this step. We will first give the experimental setup to show and discuss the results afterwards.

A. Setup

All experiment computations were performed on a single core of a Intel X5650 CPU running at 2.67 GHz. The machine was equipped with 36 GB main memory. The prediction systems were trained with up to 20,000 records of Go games played on the KGS-Go-Server[4] by strong amateur players on the 19x19 board size. The games were obtained from u-go.net[5]. The test set contained 1,000 game records from the same source, however, the test set was disjoint from the training set. We made a single harvesting run for each of three differently sized training sets (5,000, 10,000 and 20,000 game records) to produce the sets of shape patterns that were considered by the learning systems afterwards. These sets were built from all shape patterns that were seen at least 10 times during harvesting. Apart from this, a subset of 13 feature patterns from the set of feature patterns presented in Coulom [5] was used.

B. Results

We will first give some insights into the kind of shape patterns that were harvested on the training set of 20,000 game records. Fig. 4 shows the percentage of occurrences of patterns of different sizes at different phases of the games. A phase lasts for 30 moves. One can see that, in early game phases with only a few pieces on the board mostly large patterns are matched. Naturally, as the positions get more complex the more pieces are placed on the board, mostly small patterns are matched more than 10 times in later game stages.

Fig. 5 shows the cumulative distribution of finding the expert move within the first n ranked moves for the four prediction systems with a training set size of 10,000 games. The rate of ranking the expert move first is, for each system, larger as in the experimental results presented in the corresponding original papers. This might be the case because of a larger training set for MM and the use of pattern teams for the Loopy-Bayesian Ranking system. Even further, the curves are pretty similar for all systems except for the very simple probabilistic ranking.

Fig. 6 gives a closer look to the differences between the Bayesian systems by showing the distance of the cumulative distributions to the one of the probabilistic ranking system. Here we can see a small but significant better performance of the Loopy-Bayesian Ranking system and MM over Bayesian Approximation Ranking in ranking the expert move first. However, for finding the expert move in the first n ranked moves for $n \geq 3$ all systems perform almost equal.

[4]http://www.gokgs.com/

[5]http://u-go.net/gamerecords/

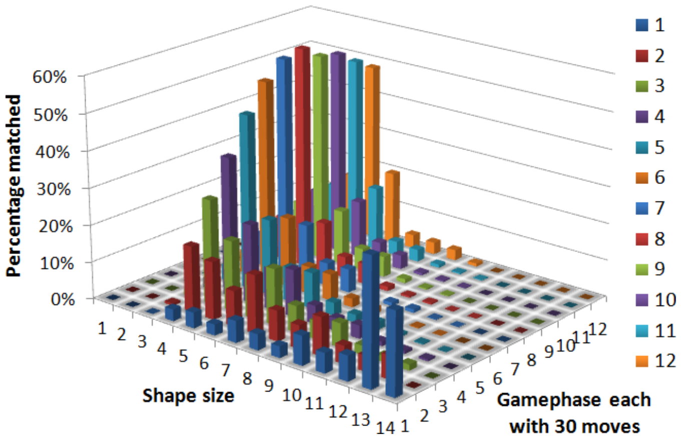


Fig. 4: Percentage of shapes of varying sizes that have been rediscovered more than 10 times in different game phases over 20,000 games.

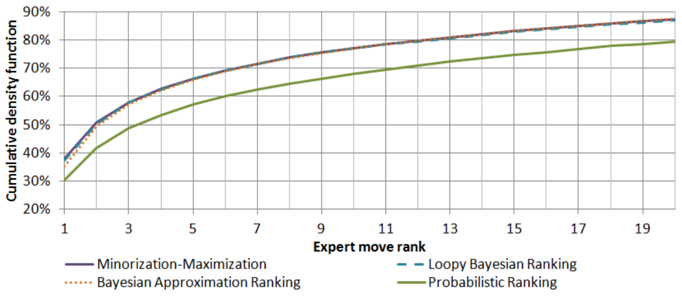


Fig. 5: Cumulative distribution of finding the expert move within the first n ranked moves.

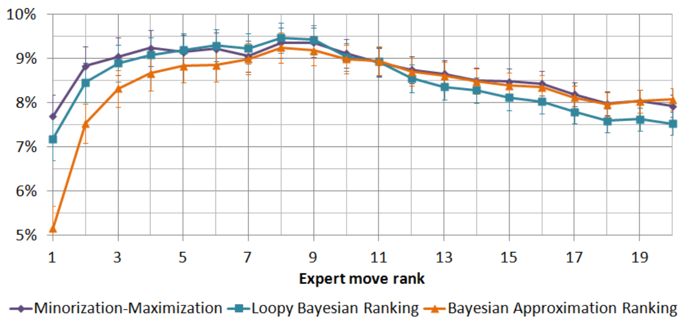


Fig. 6: Difference of cumulative prediction rank distribution to probabilistic ranking distribution. (95% conf. intervals)

Table I shows the development of prediction rates for different training set sizes. It can be seen that all systems benefit from increasing training set sizes. We could not make experiments with MM for 20.000 training games due to memory limitations.

Regarding the distributions given in Fig. 5 and 6, we should

TABLE I: Prediction rates for different training set sizes

	5,000	10,000	20,000
Minorization-Maximization	37.00%	37.86%	-
Loopy Bayesian Ranking	36.36%	37.35%	38.04%
Bayesian Approximation Ranking	34.24%	35.33%	36.19%
Probabilistic Ranking	29.16%	30.17%	30.92%

note that there are a lot more possible moves in early game stages than in the end games, when the board is almost full of pieces. Therefore we measured the rank errors of the systems at different game phases. The rank error is the fraction of the expert move rank assigned by the system over the number of available moves. Fig. 7 shows these ranking errors in a box plot. The boxes range from the lower to the upper quartile of the data and contain the median line. The whiskers cover 95% of the data. Data points outside the whiskers range are explicitly shown by single data points.

Another interesting point is the dependency of the prediction rate of expert moves to the size of the shape pattern that was matched for the particular move. The larger the pattern, the more information about the actual board configuration is encoded in the pattern, so we expect to get better prediction rates, the larger the matched patterns are. Fig. 8 shows the achieved average prediction rates for the different pattern sizes. In addition the figure gives the distribution over all matched patterns of different sizes.

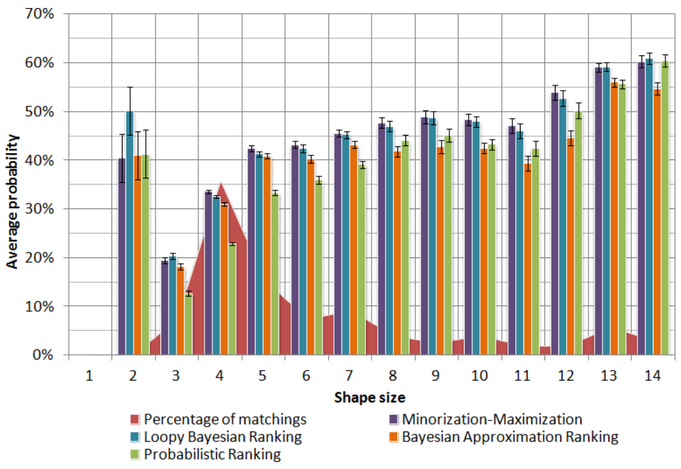


Fig. 8: Average prediction rates in relation to the size of the pattern that was matched at the expert move.

It can be seen that most of the patterns that are matched during a game are of size 3 to 5 while at the same time, pattern sizes of 3 and 4 show the significantly worst prediction rates. In order to improve here, we added a small feature vector to shape patterns of size 2 to 4. Here we took exactly the same features as done by Stern et al. [9]. Fig. 9 shows the achieved improvements for the different ranking systems at different game phases. Following Fig. 4, small shape patterns are mostly matched in later game phases, which explains the increasing impact of this modifications for later game phases.

Fig. 10 shows the average time needed by the systems

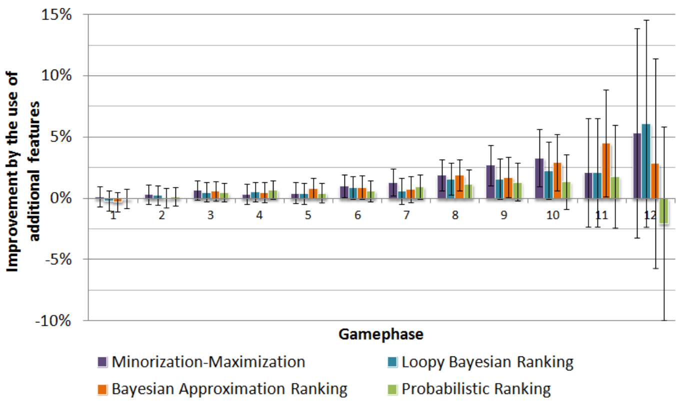


Fig. 9: Relative strength improvements from adding a small feature vector to shape patterns of size 2 to 4 at different game phases.

per game during parameter training. Not surprisingly, Loopy Bayesian Ranking required a lot more time than Bayesian Approximation Ranking. MM was iterated until convergence to a certain level, leading to varying time consumption for different training set sizes.

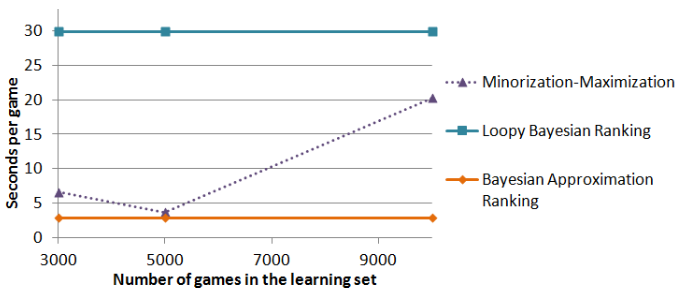
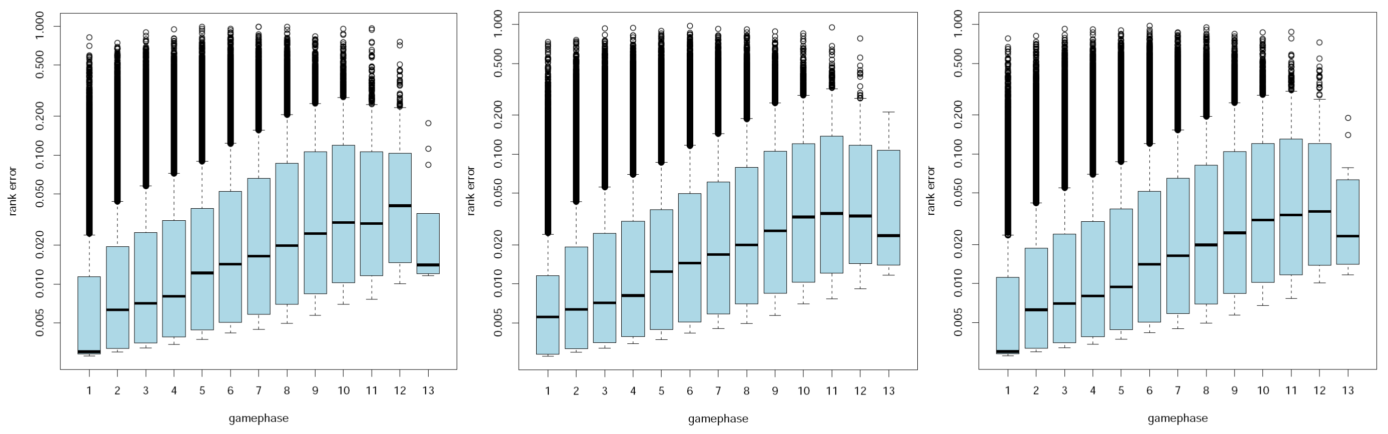


Fig. 10: Time needed per game on different training set sizes

V. Conclusion and Future Work

In this paper we compared three Bayesian systems for move prediction under fair conditions and investigated their performance by using them for move prediction with the game of Go. We observed that all prediction systems have a comparable performance concerning their prediction accuracy but differ significantly in the amount of time needed for model parameter training. Additionally we presented some insight into the use of shape patterns that are prominent for modeling in the game of Go. Following our observation, we were able to improve our pattern system further and gained some improvement in terms of prediction rate.

The outcome of the investigations presented in this paper should serve as a base for further improving the performance of state-of-the-art MCTS based Go programs by online adaptation of Monte-Carlo playouts. Recent strength improvement of Go playing programs, especially on large board sizes, were made by adding more and more domain knowledge to the playout policies that made them smarter and capable to *understand* more difficult situations that require selective and



(a) Loopy Bayesian Ranking (b) Bayesian Approximation Ranking (c) Minorization-Maximization

Fig. 7: Ranking errors at different stages of the game

well focused play. In order to further improve these policies we (among others) try to adapt playout policies to the actual board position using insights gathered by MC sampling. As a conclusion from the results presented in this paper, Bayesian Approximation Ranking appears to be a promising candidate for the use with online model adaptation in such time critical environments.

Short before submission deadline, Rémi Coulom drew our attention to a ranking system proposed by Łukasz Lew in his PhD Thesis [24], that partly builds on his approach but allows for online learning. Unfortunately, we weren't able to include this promising approach in our comparison.

Acknowledgments

The authors would like to thank Rémi Coulom and Ruby Chiu-Hsing Weng for their kind feedback as well as the anonymous reviewers who helped to further improve the quality of this paper with their comments.

References

[1] A. L. Zobrist, "A New Hashing Method with Application for Game Playing," Computer Sciences Department, University of Wisconsin, Tech. Rep. 88, 1969.

[2] ——, "Feature Extraction and Representation for Pattern Recognition and the Game of Go," Ph.D. dissertation, University of Wisconsin, Aug. 1970.

[3] S. Gelly, Y. Wang, R. Munos, and O. Teytaud, "Modifications of UCT with Patterns in Monte-Carlo Go," INRIA, Tech. Rep. 6062, 2006. [Online]. Available: http://hal.inria.fr/docs/00/12/15/16/PDF/RR-6062.pdf

[4] N. Araki, K. Yoshida, Y. Tsuruoka, and J. Tsujii, "Move Prediction in Go with the Maximum Entropy Method," in *IEEE Symposium on Computational Intelligence and Games*, Apr. 2007, pp. 189–195.

[5] R. Coulom, "Computing Elo Ratings of Move Patterns in the Game of Go," in *ICGA Journal*, vol. 30, no. 4, 2007, pp. 198–208. [Online]. Available: http://remi.coulom.free.fr/Amsterdam2007/MMGoPatterns.pdf

[6] D. Silver and G. Tesauro, "Monte-Carlo Simulation Balancing," in *International Conference on Machine Learning*, 2009, pp. 945–952. [Online]. Available: www.cs.mcgill.ca/ icml2009/papers/500.pdf

[7] S.-C. Huang, R. Coulom, and S.-S. Lin, "Monte-Carlo Simulation Balancing in Practice," in *Conference on Computers and Games*, 2010, pp. 81–92. [Online]. Available: /home/slars/projects/paper/Hu10.pdf

[8] S. Gelly and D. Silver, "Combining Online and Offline Knowledge in UCT," in *International Conference on Machine Learning*, 2007, pp. 273–280.

[9] D. Stern, R. Herbrich, and T. Graepel, "Bayesian Pattern Ranking for Move Prediction in the Game of Go," in *Proceedings of the International Conference of Machine Learning*, Jan. 2006. [Online]. Available: http://research.microsoft.com/pubs/67955/p873-stern.pdf

[10] R. C. Weng and C.-J. Lin, "A Bayesian Approximation Method for Online Ranking," *Journal of Machine Learning Research*, vol. 12, pp. 267–300, Jan. 2011.

[11] J. Tromp and G. Farnebäck, "Combinatorics of Go," in *Proc. on the Int. Conf. on Computers and Games*. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 84–99. [Online]. Available: http://dl.acm.org/citation.cfm?id=1777826.1777834

[12] C. Donninger and U. Lorenz, "The Chess Monster Hydra," in *Proc. of Int. Conf. on Field-Programmable Logic and Applications (FPL)*, ser. LNCS, vol. 3203, 2004, pp. 927–932.

[13] U. Lorenz, "Parallel Controlled Conspiracy Number Search," in *Proc. of Int. Euro-Par Conf. (Euro-Par)*, ser. LNCS, vol. 2400, 2002, pp. 420–430.

[14] A. Rimmel, O. Teytaud, C.-S. Lee, S.-J. Yen, M.-H. Wang, and S.-R. Tsai, "Current Frontiers in Computer Go," in *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 2, no. 4, Dec. 2010, pp. 229–238.

[15] C. Browne, E. Powley, D. Whitehouse, S. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A Survey of Monte Carlo Tree Search Methods," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 4, no. 1, pp. 1–43, Mar. 2012.

[16] D. Stoutamire, "Machine learning, game play, and Go," Case Western Reserve University, Tech. Rep. 91-128, 1991.

[17] R. A. Bradley and M. E. Terry, "Rank Analysis of Incomplete Block Designs: I. The Method of Paired Comparisons," *Biometrika*, vol. 39, no. 3/4, pp. 324–345, Dec. 1952. [Online]. Available: http://www.jstor.org/stable/2334029

[18] A. E. Elo, *The Rating of Chessplayers, Past and Present*. New York: Arco Publishing, 1986.

[19] D. R. Hunter, "MM algorithms for generalized Bradley Terry models," *The Annals of Statistics*, vol. 32, no. 1, pp. 384–406, 2004.

[20] D. J. C. MacKay, *Information Theroy, Inference, and Learning Algorithms*. Cambridge University Press, 2003, ch. IV-26: Exact Marginalization in Graphs.

[21] T. Minka, "A family of algorithms for approximate Bayesian inference," Ph.D. dissertation, Massachusettes Institute of Technology, 2001.

[22] R. Herbrich, T. Minka, and T. Graepel, "TrueSkill(TM): A Bayesian skill rating system," in *Advances in Neural Information Processing Systems 20*. MIT Press, 2007, pp. 569–576.

[23] D. Stern, "Modelling Uncertainty in the Game of Go," Ph.D. dissertation, University of Cambridge, Feb. 2008.

[24] L. Lew, “Modeling Go Game as a Large Decomposable Decision Process,” Ph.D. dissertation, Warsaw University, June 2011.