Integrating Factorization Ranked Features in MCTS: an Experimental Study

Chenjun Xiao and Martin Müller

Computing Science, University of Alberta
Edmonton, Canada
{chenjun,mmueller}@ualberta.ca

Abstract. Recently, *Factorization Bradley-Terry (FBT)* model is introduced for fast move prediction in the game of Go. It has been shown that FBT outperforms the state-of-the-art fast move prediction system of Latent Factor Ranking (LFR). In this paper, we investigate the problem of integrating feature knowledge learned by FBT model in Monte Carlo Tree Search. We use the open source Go program Fuego as the test platform. Experimental results show that the FBT knowledge is useful in improving the performance of Fuego.

1 Introduction

The idea of Monte Carlo Tree Search (MCTS) [2] is to online construct a search tree of game states evaluated by fast Monte Carlo simulations. However in games with large state space, accurate value estimation by simple simulation cannot be easily guaranteed given limited search time. The inaccurate estimation can mislead the growth of the search tree and can severely limit the strength of the program. Thereby, it's reasonable to incorporate the domain knowledge of the game to serve as a heuristic information that benefits the search.

In Computer Go [10] research, knowledge is usually represented by features, such as shape patterns and tactical features. A *move prediction system* applies machine learning techniques to acquire the feature knowledge from professional game records or self played games. Selective search algorithm can then focus on the most promising moves evaluated by such system [1]. For example, [3] proposes Minorization-Maximization (MM) to learn feature knowledge offline and uses it to improve random simulation. [7] considers feature knowledge as a prior to initial statistical values when a new state is added to the search tree. AlphaGo [12] incorporates supervised learned Deep Convolutional Neural Networks (DCNN) as part of in-tree policy for move selection, and further improve the network with reinforcement learning from games of self-play to get a powerful value estimation function. The integrated system becomes the first program to ever beat the world's top Go player.

Recently, [15] introduces *Factorization Bradley-Terry (FBT)* model to learn feature knowledge which became the state-of-the-art fast move prediction algorithm. The major innovation of FBT model is to consider the interaction between different features as part of a probability-based framework, which can be considered as a combination of two leading approaches: MM [3] and LFR [14]. However, it's still not clear whether the feature knowledge learned by this model is useful to improve the strength of the MCTS framework. We investigate this problem in this paper by integrating FBT based knowledge in the open source program Fuego [5].

The remaining of this paper is organized as follows: Section 2 describes the idea of FBT model for move prediction in Go. Section 3 discusses how to integrate FBT based knowledge within MCTS framework. Section 4 describes the feature knowledge and move selection scheme in current Fuego and provides the experimental results. Section 5 gives a conclusion of this work and discusses the possible future work.

2 Factorization Bradley-Terry Model for Move Prediction Problem

We briefly describe how FBT model works for move prediction problem in the game of Go. In most popular high-speed move prediction systems, each move is represented as a combination of a group of features. Weights for each feature are learned from expert game records by supervised learning algorithms, and an evaluation function based on the weights is defined to rank moves.

Specifically, let $\mathcal{S}$ be the set of possible Go positions, $\mathit{\Gamma}(s)$ be the set of legal moves in a specific position $s \in \mathcal{S}$, and $\mathcal{F}$ be the set of features which are used to describe moves in a given game state. Each move is represented by its set of active features $\mathcal{G} \subseteq \mathcal{F}$. The training set $\mathcal{D}$ consists of cases $\mathcal{D}_j$, with each case representing the possible move choices in one game position s_j, and the expert move is specified as $\mathcal{G}_j^*$.

$$\mathcal{D}_j = \{\ \mathcal{G}_j^i \mid for\ i = 1, \ldots, |\mathit{\Gamma}(s_j)|\}$$

Most high-speed move prediction systems usually differ from the method of predicting $\mathcal{G}_j^*$ from $\mathcal{D}_j$ as well as the model of the strength of $\mathcal{G}$. In MM [3], the strength of a group $\mathcal{G}$ is approximated by the sum of weights of all features within the group. Prediction of $\mathcal{G}_j^*$ is formulated as a competition among all possible groups. A simple probabilistic model named Generalized Bradley-Terry model [8] defines the probability of each feature group winning a competition. While in another efficient move prediction algorithm called Latent Factor Ranking (LFR) [14], the strength of a group is modelled using a Factorization Machine (FM), which also takes pairwise interactions between features into account besides the sum of all features' weights. Prediction of $\mathcal{G}_j^*$ is simply formulated as a binary classification problem, with $\mathcal{G}_j^*$ in one class and all other groups in the other. LFR outperforms MM in terms of move prediction accuracy. But the evaluation function LFR produces does not provide a probability distribution over all possible moves, which makes it much harder to combine with other kinds of knowledge.

FBT model takes advantage of both MM and LFR: it considers the interaction between features within a group, and produce the evaluation function in a probability-based framework. In FBT, the strength of a group $\mathcal{G} \subseteq \mathcal{F}$ is defined in a same way as in LFR

$$E_{\mathcal{G}} = \sum_{f \in \mathcal{G}} w_f + \frac{1}{2} \sum_{f \in \mathcal{G}} \sum_{g \in \mathcal{G}, g \neq f} \langle v_f, v_g \rangle$$

where $w_f \in \mathbb{R}$ is the (estimated) *strength*, and $v_f \in \mathbb{R}^k$ is the *factorized interaction vector*, of a feature $f \in \mathcal{F}$. The *interaction strength* between two features f and g is modeled as $\langle v_f, v_g \rangle = \sum_{i=1}^{k} v_{f,i} \cdot v_{g,i}$, where k is the pre-defined dimension of the factorization. In Computer Go, setting $k = 5$ and $k = 10$ are most popular [14]. Richer feature sets might require larger k for best performance. With the definition of $E_{\mathcal{G}}$, FBT then applies the Generalized Bradley-Terry model for each test case $\mathcal{D}_j$,

$$P(\mathcal{D}_j) = \frac{exp(E_{\mathcal{G}_j^*})}{\sum_{i=1}^{|\Gamma(s_j)|} exp(E_{\mathcal{G}_j^i})}$$

Suitable parameters in FBT are estimated by maximizing the likelihood of the training data, using a Stochastic Gradient Decent (SGD) algorithm. [15] also provides two techniques to accelerate the training process: an efficient incremental implementation of gradient update, and an unbiased approximate gradient estimator. Details of these two techniques as well as the induction of parameter update formula can be found in [15].

3 Integrating FBT Knowledge in MCTS

As suggested before, a move prediction system can provide useful initial recommendations of which moves are likely to be the best. Selective search with proper exploration scheme, such as MCTS, can further improve upon these recommendations with online simulation information. One favourable property of FBT model is to produce a probability based evaluation. Intuitively, it's a probability distribution of which move is going to be selected by a human expert under a game state. Therefore, it seems very straightforward to incorporate FBT knowledge as part of exploration, since we should explore more on moves which are most favoured by human experts.

We apply a variant of PUCT [11] formula which is used in AlphaGo [12] to integrate FBT knowledge in MCTS. The idea of this formula is to explore moves according to a value that is proportional to the predicted probability but decays with repeated visits as in original UCT style [9]. When a new game state s is added to the search tree, we call a pre-trained FBT model to get a prediction $P_{FBT}(s, a)$, which assigns an exploration bonus $E_{FBT}(s, a)$ for each move $a \in \Gamma(s)$. In order to keep sufficient exploration, we set a lower cut threshold λ_{FBT}, where for all $a \in \Gamma(s)$ if $P_{FBT}(s, a) < \lambda_{FBT}$ then simply let $E_{FBT}(s, a) = \lambda_{FBT}$, otherwise $E_{FBT}(s,a) = P_{FBT}(s,a)$. At state s during in-tree move selection, the algorithm will select the move

$$a = \text{argmax}_{a'}(Q(s,a') + c_{puct}E_{FBT}(s,a')\sqrt{\frac{\lg(N(s))}{1+N(s,a')}}) \qquad (1)$$

where $Q(s,a)$ is the accumulated move value estimated by online simulation, c_{puct} is a exploration constant, $N(s,a)$ is the number of visit time of move a in s, and $N(s) = \sum_i N(s,i)$.

4 Experiments

We use the open source program Fuego [5] as our experiment platform to test if FBT knowledge is helpful for improving MCTS. We first introduce the feature knowledge in current Fuego system, then introduce the training settlement for the FBT model and the setup for the experiment, and finally present the results.

4.1 Feature Knowledge for Move Selection in Fuego

Prior Feature Knowledge The latest Fuego (svn version 2017) applies feature knowledge to initialize statistical information when a new state is added to the search tree. A set of features trained with LFR [14] is used where interaction dimension is set at $k = 10$. Since the evaluation LFR produces is a real value indicating the strength of the move without any probability based interpretation, Fuego designed a well-tuned formula to transfer the output value to the prior knowledge for initialization. It adopts a similar method as suggested in [7], where the prior knowledge contains two parts: $N_{prior}(s,a)$ and $Q_{prior}(s,a)$. This indicates that MCTS would perform $N_{prior}(s,a)$ simulations to achieve an estimate of $Q_{prior}(s,a)$ accuracy. Let $V_{LFR}(s,a)$ be the evaluation of move $a \in \Gamma(s)$, $V_{largest}$ and $V_{smallest}$ be the largest and smallest evaluated value respectively. Fuego uses the following formula to assign $N_{prior}(s,a)$ and $Q_{prior}(s,a)$,

$$N_{prior}(s,a) = \begin{cases} \frac{c_{LFR}*|\Gamma(s)|}{SA} * V_{LFR}(s,a) & \text{if } V_{LFR}(s,a) \geq 0 \\ -\frac{c_{LFR}*|\Gamma(s)|}{SA} * V_{LFR}(s,a) & \text{if } V_{LFR}(s,a) < 0 \end{cases} \qquad (2)$$

$$Q_{prior}(s,a) = \begin{cases} 0.5 * (1 + V_{LFR}(s,a)/V_{largest}) & \text{if } V_{LFR}(s,a) \geq 0 \\ 0.5 * (1 - V_{LFR}(s,a)/V_{smallest}) & \text{if } V_{LFR}(s,a) < 0 \end{cases} \qquad (3)$$

where $SA = \sum_i |V_{LFR}(s,i)|$ is the sum of absolute value of each move's evaluation. When a new game state is added to the search tree, Fuego will call the method showed above to initialize the state's statistical information by setting $N(s,a) \leftarrow N_{prior}(s,a)$ and $Q(s,a) \leftarrow Q_{prior(s,a)}$.

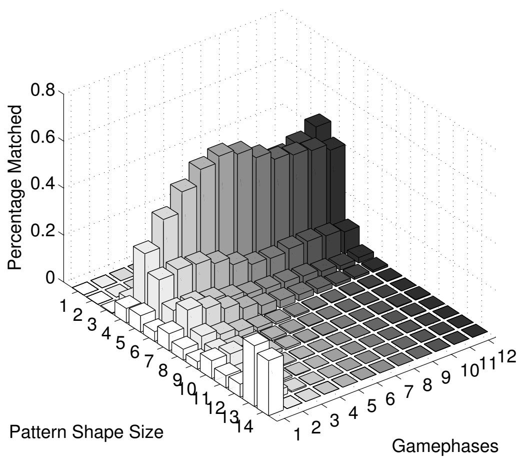

Fig. 1. Distribution of patterns with different size harvested at least 10 times in different game phases.

Greenpeep Knowledge Another kind of knowledge Fuego also has as part of in-tree move selection policy is called Greenpeep Knowledge. It uses a predefined table to get a probability based knowledge $P_g(s,a)$ about each move $a \in \Gamma s$. Then the knowledge is added as a bias for move selection according to the PUCT formula [11]. The reason why Fuego does not use LFR knowledge to replace Greenpeep knowledge might be that LFR cannot produce probability based evaluation. Details can be found in the Fuego source code base [4].

Move Selection in Fuego In summary, Fuego adopts the following formula to select moves during in-tree search,

$$a = \text{argmax}_{a'}(Q(s,a') - \frac{c_g}{\sqrt{P_g(s,a')}} \times \sqrt{\frac{N(s,a')}{N(s,a')+5}}) \tag{4}$$

where c_g is a parameter controlling the scale of the Greenpeep knowledge, . $Q(s,a')$ is initialized according to equation (2) and (3), and further improved with Monte Carlo simulation and Rapid Action Value Estimation (RAVE). Note that formula (4) does not have the UCB style exploration term, since the exploration constant is set to zero in Fuego. The only exploration comes from RAVE. Comparing formula (4) with (1), we could consider the FBT knowledge $P_{FBT}(s,a)$ as a replacement of the Greenpeep knowledge $P_g(s,a)$, but with a different way to be added as a bias and different decay function.

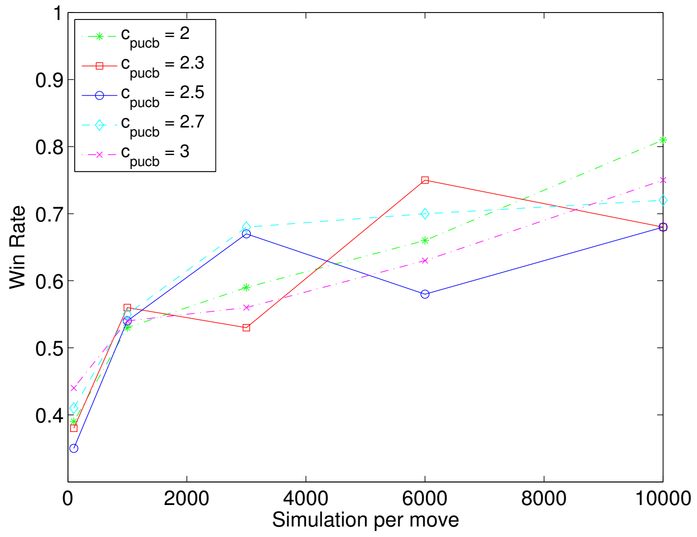

Fig. 2. Experimental Results: FBT-FuegoNoLFR vs FuegoNoLFR.

4.2 Training Settlement for FBT Model

We train a FBT model with interaction dimension $k = 5$ using 10000 master games download from the public domain at `https://badukmovies.com/pro_games`. The prediction accuracy of this model is 38.26%. The parameters of the training algorithm including learning rate and regularization parameters are set at the same as described in [15]. We also apply the same stopping criteria that the training process is stopped and the best performing weight set is returned if the prediction accuracy on a validation set does not increase for three iterations.

The simple features used in this work are listed below. Most features are the same as suggested in [15]. We only use large pattern as the shape pattern for this experiment. All patterns are harvested as in [13, 14]. Figure (1) shows the distribution of harvested largest matches for the different pattern sizes in each game phase. The implementation of the tactical features is part of the Fuego program [5], details can be found in the Fuego code base [4]. Note that current Fuego includes the same set of tactical features. But it uses small shape patterns instead of large patterns for feature knowledge evaluation.

- **Pass**
- **Capture, Extension, Self-atari, Atari** Tactical features similar to [3].
- **Line and Position (edge distance perpendicular to Line)** ranges from 1 to 10.
- **Distance to previous move** feature values are 2,..., 16, $\geq$ 17. The distance is measured by $d(\delta x, \delta y) = |\delta x| + |\delta y| + max\{|\delta x|, |\delta y|\}$.

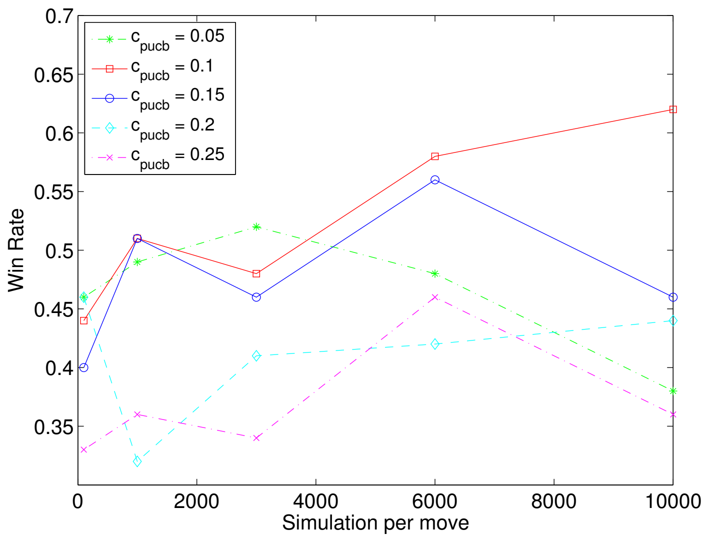

Fig. 3. Experimental Results: FBT-Fuego vs Fuego.

- **Distance to second-last move** uses the same metric as previous move. The distance can be 0.
- **Fuego Playout Policy** These features correspond to the rules in the playout policy used in Fuego. Most are simple tactics related to stones with a low number of liberties and 3x3 patterns.
- **Side Extension** The distance to the closest stones along the sides of the board.
- **Corner Opening Move** Standard opening moves.
- **CFG Distance** Distance when contracting all stones in a block to a single node in a graph [6].
- **Shape Patterns** Circular patterns with sizes from 2 to 14. All shape patterns are invariant to rotation, translation and mirroring.

4.3 Setup

Experiments are performed on a 2.4 GHz Intel Xeon CPU with 64 GB memory. We use the latest Fuego (svn revision 2017) in the experiment. We call Fuego without LFR prior knowledge as FuegoNoLFR, Fuego applying formula (1) to select moves in-tree as FBT-Fuego, and Fuego without LFR but using formula (1) as FBT-FuegoNoLFR. The lower cut threshold for FBT knowledge is set to $\lambda_{FBT} = 0.001$. All other parameters are default as in the original settings of Fuego.

Program Name	100	1000	3000	6000	10000
FBT-FuegoNoLFR	11.8	192.4	704.1	1014.5	1394.9
FuegoNoLFR	5.1	55.7	148.7	225.2	354.4
FBT-Fuego	23.4	241.1	734.1	912.6	1417.5
Fuego	10.8	168.3	564.2	778.6	1161.2

Table 1. Running time comparison (specified in seconds) with different simulations for per move.

4.4 Experimental Results

We first compare FBT-FuegoNoLFR with FuegoNoLFR. This experiment is designed to show the strength of FBT knowledge without any influence from other kinds of knowledge. We test the performance of FBT-FuegoNoLFR against FuegoNoLFR with different exploration constants c_{puct}. After initial experiments, the range explored was $c_{puct} \in \{2, 2.3, 2.5, 2.7, 3\}$. In order to investigate if the FBT knowledge is scaling with the number of simulations per move, N_{sim} was tested by setting $N_{sim} \in \{100, 1000, 3000, 6000, 10000\}$. Figure 2 shows the win rate of FBT-FuegoNoLFR against FuegoNoLFR. All data points are averaged over 1000 games. The results show that adding FBT knowledge can dramatically improve the performance of Fuego over the baseline without feature knowledge as prior. FBT-FuegoNoLFR scales well with more simulations per move. With $c_{pucb} = 2$ and 10000 simulations per move FBT-FuegoNoLFR can beat FuegoNoLFR with 81% winning rate.

We then compare FBT-Fuego with full Fuego, in order to investigate if the FBT knowledge is comparable with current feature knowledge in Fuego and able to improve the performance in general. In this case, c_{puct} is tuned over a different range, $c_{puct} \in \{0.05, 0.1, 0.15, 0.2, 0.25\}$. $N_{sim} \in \{100, 1000, 3000, 6000, 10000\}$, and all data points are averaged over 1000 games as before. Results are presented in Figure 3. FBT-Fuego has worse performance in most settings of c_{puct}. But it can be made to work after careful tuning. As suggested in Figure 3, under the setting where $c_{pucb} = 0.1$, FBT-Fuego scales well with the number of simulations per move, and achieves 62% winning rate against Fuego with 10000 simulations per move. One possible reason is that the FBT knowledge is not quite comparable with the LFR knowledge. The moves these two methods favour might be different in some situations, which makes it very hard to tune a well tuned system when adding another knowledge term.

Finally, we show the running time of our methods with different simulations per move in Table 1. FBT-FuegoNoLFR spends much more time than FuegoNoLFR, since FuegoNoLFR only uses Greenpeep knowledge for exploration and thus does not need to compute any feature knowledge. FBT-FuegoNoLFR spends a little less time than FBT-Fuego, since it does not use feature knowledge to initialize prior knowledge. The speed of FBT-Fuego is a little worse than Fuego. The time difference is spent on computing large patterns, while Fuego only uses small shape patterns.

5 Conclusion and Future Work

In this paper, we introduce how to integrate the state-of-the-art fast move prediction algorithm FBT in MCTS. We use the open source program Fuego as our test platform. Experimental results show that FBT knowledge is useful to improve the performance of Fuego, without too much sacrifice in efficiency.

Future work includes: 1. try to discover a method to transform FBT knowledge as prior knowledge for initialization. 2. try to apply the FBT knowledge for improving fast roll-out policy, which has been shown as a very important part in the state-of-the-art Go program AlphaGo [12].

References

1. C. Browne, E. Powley, D. Whitehouse, S. Lucas, P. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton. A survey of Monte Carlo tree search methods. *IEEE Trans. Comput. Intellig. and AI in Games*, 4(1):1–43, 2012.
2. R. Coulom. Efficient selectivity and backup operators in Monte-Carlo tree search. In J. van den Herik, P. Ciancarini, and H. Donkers, editors, *Proceedings of the 5th International Conference on Computer and Games*, volume 4630/2007 of *Lecture Notes in Computer Science*, pages 72–83, Turin, Italy, June 2006. Springer.
3. Rémi Coulom. Computing Elo ratings of move patterns in the game of Go. In *Proc. Computer Games Workshop 2007 (CGW2007)*, pages 113–124. 2007.
4. M. Enzenberger and M. Müller. Fuego, 2008-2015. `http://fuego.sourceforge.net`.
5. M. Enzenberger, M. Müller, B. Arneson, and R. Segal. Fuego - an open-source framework for board games and Go engine based on Monte Carlo tree search. *IEEE Transactions on Computational Intelligence and AI in Games*, 2(4):259–270, 2010.
6. K. J. Friedenbach. *Abstraction Hierarchies: A Model of Perception and Cognition in the Game of Go.* PhD thesis, University of California, Santa Cruz, 1980.
7. S. Gelly and D. Silver. Combining online and offline knowledge in UCT. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 273–280. ACM, 2007.
8. David R Hunter. MM algorithms for generalized Bradley-Terry models. *Annals of Statistics*, pages 384–406, 2004.
9. L. Kocsis and C. Szepesvári. Bandit based Monte-Carlo planning. In J. Fürnkranz, T. Scheffer, and M. Spiliopoulou, editors, *Machine Learning: ECML 2006*, volume 4212 of *Lecture Notes in Computer Science*, pages 282–293. Springer Berlin / Heidelberg, 2006.
10. M. Müller. Computer Go. *Artificial Intelligence*, 134(1–2):145–179, 2002.
11. C. Rosin. Multi-armed bandits with episode context. *Ann. Math. Artif. Intell.*, 61(3):203–230, 2011.
12. David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.
13. David Stern, Ralf Herbrich, and Thore Graepel. Bayesian pattern ranking for move prediction in the game of Go. In *Proceedings of the 23rd international conference on Machine learning*, pages 873–880. ACM, 2006.

14. M. Wistuba and L. Schmidt-Thieme. Move prediction in Go - modelling feature interactions using latent factors. In I. Timm and M. Thimm, editors, *KI2013*, volume 8077 of *Lecture Notes in Computer Science*, pages 260–271. Springer, 2013.
15. Chenjun Xiao and Martin Müller. Factorization ranking model for move prediction in the game of go. In *AAAI*, 2016.