Computing Elo Ratings of Move Patterns in the Game of Go

Paper by Rémi Coulom, CG 2007

Presented by Markus Enzenberger.
Go Seminar, University of Alberta.

May 6, 2007

Outline

Introduction

Minorization-Maximization / Bradley-Terry Models

Experiments in the Game of Go

Usage in a MC-Program

Conclusion

Introduction

- Patterns are useful for Go programs
  - Prune search trees
  - Order moves
  - Improve random simulations in Monte-Carlo programs
- One approach for learning patterns:
  Extract frequent patterns from expert games
- New supervised learning algorithm
  based on Bradley-Terry model (theoretical basis of Elo system)

Elo rating system

- Assign numerical strength value to players
- Compute strength from game results
- Estimates a probability distribution for future game results

Apply to move patterns

- Each move is a victory of one pattern over the others
- Elo ratings give a probability distribution over moves

Related Work

Simplest approach: Measure frequency of play of each pattern
(Bouzy/Chaslot 2005) (Moyo Go Studio)

$$\text{Rating(Pattern)} = \frac{\text{number of times played}}{\text{number of times present}}$$

- Stronger patterns are played sooner $\rightarrow$ higher rating
- Does not take strength of competing patterns into account
(Elo-rating analogy: measure only winning rate
independent of opponent strength)

Bayesian pattern ranking

(Stern/Herbrich/Graepel 2006)

- Takes strength of opponents into account
- Patterns to evaluate grows exponentially with number of features
- Restricted to only a few move features

Maximum-entropy classification

(Araki/Yoshida/Tsuruoka/Tsujii 2007)

- Addresses the problem of combining move features
- Does not take strength of opponents into account
- High computational cost

Minorization-Maximization / Bradley-Terry Models

Introduction

Minorization-Maximization / Bradley-Terry Models
- Elo Ratings and the Bradley-Terry Model
- Generalizations of the Bradley-Terry Model
- Relevance of the Bradley-Terry Model
- Bayesian Inference
- Minorization-Maximization

Experiments in the Game of Go

Usage in a MC-Program

Elo Ratings and the Bradley-Terry Model

- γ_i is a (positive) value for the strength of individual i

Estimation fo the probability that i beats j:

$$P(i \text{ beats } j) = \frac{\gamma_i}{\gamma_i + \gamma_j}$$

(Elo rating of i is defined by $r_i = 400 \log_{10}(\gamma_i)$)

Generalizations of the Bradley-Terry Model

Competitions between more than one individual:

$$\forall i \in \{1, \ldots, n\}, P(i \text{ wins}) = \frac{\gamma_i}{\gamma_1 + \gamma_2 + \ldots + \gamma_n}$$

Competitions between teams:

$$P(\text{1-2-3 wins against 4-2 and 1-5-6-7}) = \frac{\gamma_1\gamma_2\gamma_3}{\gamma_1\gamma_2\gamma_3 + \gamma_4\gamma_2 + \gamma_1\gamma_5\gamma_6\gamma_7}$$

(Hunter 2004)

Relevance of the Bradley-Terry Model

- Strong assumptions about what is being modeled
- No cycles
- Strength of a team is the sum of its members (in Elo ratings)

Bayesian Inference

The values γ_i have to be estimated from past results $\mathbf{R}$ using Bayesian inference:

$$P(\gamma|\mathbf{R}) = \frac{P(\mathbf{R}|\gamma)P(\gamma)}{P(\mathbf{R})}$$

- Find γ^* that maximizes $P(\gamma|\mathbf{R})$
- Convenient way to choose a prior distribution $P(\gamma)$ by virtual game results $\mathbf{R}'$: $P(\gamma) = P(\mathbf{R}'|\gamma)$
  $\rightarrow$ maximize $P(\mathbf{R}, \mathbf{R}'|\gamma)$

Minorization-Maximization

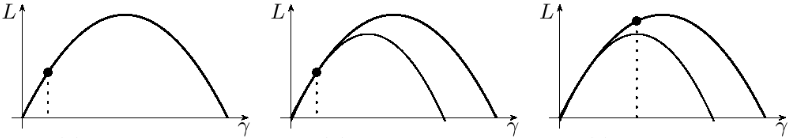
Notation

- n individuals with unknown strengths $\gamma_1, \ldots, \gamma_n$
- N results $R_1, \ldots, R_N$
- Probability of one result R_j as a function of γ_i:

$$P(R_j) = \frac{A_{ij}\gamma_i + B_{ij}}{C_{ij}\gamma_i + D_{ij}}$$

  $A_{ij}, B_{ij}, C_{ij}, D_{ij}$ do not depend on γ_i. Either A_{ij} or B_{ij} is 0.
- Objective to maximize:

$$L(\gamma_i) = \prod_{j=1}^{N} P(R_j)$$



(a) Initial guess. (b) Minorization. (c) Maximization.

- Make inital guess γ^0
- Find function m that minorizes L at γ^0
  - $m(\gamma^0) = L(\gamma^0) \quad \forall\gamma : m(\gamma) \leq L(\gamma)$
- Compute maximum γ^1 of m
- γ^1 is an improvement over γ^0

Function to be maximized

$$L(\gamma_i) = \prod_{j=1}^{N} \frac{A_{ij}\gamma_i + B_{ij}}{C_{ij}\gamma_i + D_{ij}}$$

Take logarithm:

$$\log L(\gamma_i) = \sum_{j=1}^{N} \log(A_{ij}\gamma_i + B_{ij}) - \sum_{j=1}^{N} \log(C_{ij}\gamma_i + D_{ij})$$

Define number of wins: $W_i = |\{j|A_{ij} \neq 0\}|$
Remove terms that do not depend on γ_i

$$f(\gamma_i) = W_i \log \gamma_i - \sum_{j=1}^{N} \log(C_{ij}\gamma_i + D_{ij})$$

Logarithms can be minorized by their tangent at x_0:

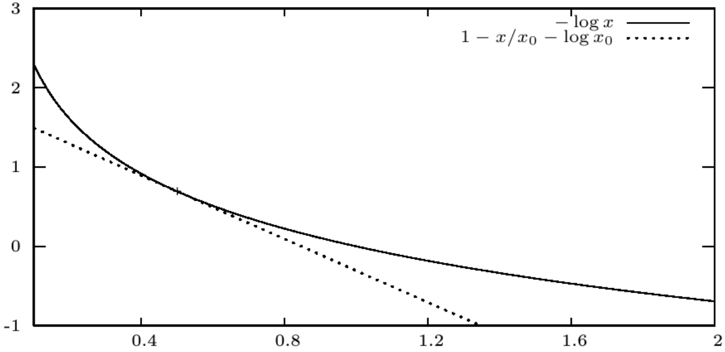


Fig. 2. Minorization of $-\log x$ at $x_0 = 0.5$ by its tangent.

Minorizing function to be maximized becomes:

$$m(\gamma_i) = W_i \log \gamma_i - \sum_{j=1}^{N} \frac{C_{ij}\gamma_i}{C_{ij}\gamma_i + D_{ij}}$$

Maximum of m is at:

$$\gamma_i = \frac{W_i}{\sum_{j=1}^{N} \frac{C_{ij}}{C_{ij}\gamma_i + D_{ij}}}$$

Minorization-Maximization Formula:

$$\gamma_i \leftarrow \frac{W_i}{\sum_{j=1}^{N} \frac{C_{ij}}{C_{ij}\gamma_i + D_{ij}}}$$

- A win counts more if
  - team mates are weak (C_{ij})
  - overall strength of participants is high ($C_{ij}\gamma_i + D_{ij}$)
- Updates can be done
  - one γ_i at a time
  - in batches (only for mutually exclusive features)

Experiments in the Game of Go

Introduction

Minorization-Maximization / Bradley-Terry Models

Experiments in the Game of Go
- Data
- Features
- Prior
- Results
- Discussion

Usage in a MC-Program

- Each position of a game is a competition
- The played move is the winner
- Each move is a team of features

Data

- Game records by strong players on the KGS Go server
- Either one player is 7d or stronger or both are 6d
- Training set: 652 games (131,939 moves)
- Test set: 551 games (115,832 moves)

Features

- Tactical features
  1. pass
  2. capture
  3. extension
  4. self-atari
  5. atari
  6. distance to border
  7. distance to previous move
  8. distance to move before previous move
- Monte-Carlo owner (63 random games)
- Shape patterns
  (16,780 shapes of radius 3–10 that occur at least 5000 times in training set)

Prior

- Virtual opponent with $\gamma = 1$
- Add one virtual win and one virtual loss against the virtual opponent for each feature
- In Elo-rating, this corresponds to a symmetric probability distribution with mean 0 and standard deviation 302

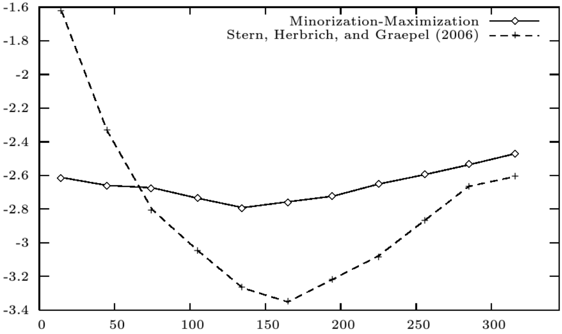
Results

Feature	Level	γ	Description
Pass	1	0.17	Previous move is not a pass
	2	24.37	Previous move is a pass
Capture	1	30.68	String contiguous to new string in atari
	2	0.53	Re-capture previous move
	3	2.88	Prevent connection to previous move
	4	3.43	String not in a ladder
	5	0.30	String in a ladder
Extension	1	11.37	New atari, not in a ladder
	2	0.70	New atari, in a ladder
Self-atari	1	0.06	
Atari	1	1.58	Ladder atari
	2	10.24	Atari when there is a ko
	3	1.70	Other atari
Distance to border	1	0.89	
	2	1.49	
	3	1.75	
	4	1.28	

Results

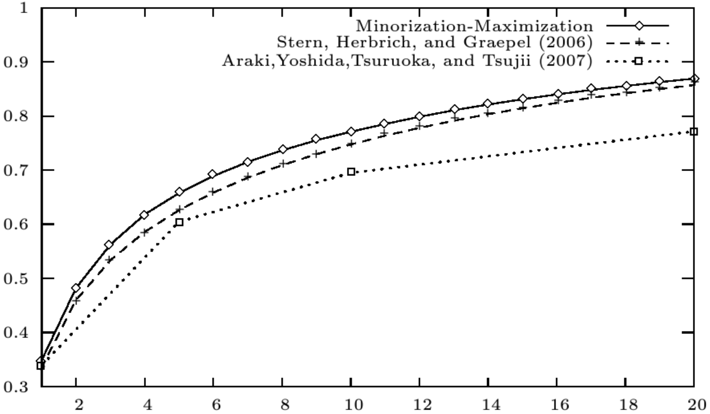
Distance to previous move	2	4.32	$d(\delta x, \delta y) = \lvert\delta x\rvert + \lvert\delta y\rvert + \max(\lvert\delta x\rvert, \lvert\delta y\rvert)$
	3	2.84	
	4	2.22	
	5	1.58	
	...	...	
	16	0.33	
	≥ 17	0.21	
Distance to the move before the previous move	2	3.08	
	3	2.38	
	4	2.27	
	5	1.68	
	...	...	
	16	0.66	
	≥ 17	0.70	
MC Owner	1	0.04	$0 - 7$
	2	1.02	$8 - 15$
	3	2.41	$16 - 23$
	4	1.41	$24 - 31$
	5	0.72	$32 - 39$
	6	0.65	$40 - 47$
	7	0.68	$48 - 55$
	8	0.13	$56 - 63$

Mean log-evidence per game stage



- Mean logarithm of probability of selecting the target move
- Better in the middle and endgame, worse in the beginning (but Stern/Herbrich/Graepel used 12,000,000 shape patterns)

Probability of finding the target move within *n* best moves



Discussion

- Best result among results published in academic papers (De Groot (Moyo Go Studio) claims 42 %
  not backed by publication)
- Used much less games (652) and shape patterns (16,780) than Stern/Herbrich/Graepel
  (181,000 games; 12,000,000 shape patterns)
- Training took only 1 hour CPU time and 600 MB RAM

Usage in a MC-Program

Introduction

Minorization-Maximization / Bradley-Terry Models

Experiments in the Game of Go

Usage in a MC-Program
- Random Simulations
- Progressive Widening
- Performance against GNU Go

Conclusion

Random Simulations

- Patterns provide probability distributions for random games
- Only fast, lightweight features
  - 3×3 shapes
  - extension (without ladder knowledge)
  - capture (without ladder knowledge)
  - self-atari
  - contiguity to previous move
- Contiguity to previous move is a strong feature
  Produces sequences of contiguous moves like in MoGo

Progressive Widening

- Crazy Stone uses patterns to prune the search tree
- Full set of features

1. Node in search tree is first searched for a while with random simulations
2. Then node is promoted to internal node and pruning is applied

Pruning algorithm:

Restrict search to first n node, with n growing with the logarithm of number of simulations:
add n^{th} node ($n \geq 2$) after $40 \times 1.4^{n-2}$ simulations

- Due to strength of contiguity feature, this tends to produce a local search

Performance against GNU Go

Pat.	P.W.	Size	Min./game	GNU Level	Komi	Games	Win ratio
-	-	9×9	1.5	10	6.5	170	38.2%
x	-	9×9	1.5	10	6.5	170	68.2%
x	x	9×9	1.5	10	6.5	170	90.6%
-	-	19×19	32	8	6.5	192	0.0%
x	-	19×19	32	8	6.5	192	0.0%
x	x	19×19	32	8	6.5	192	37.5%
x	x	19×19	128	8	6.5	192	57.1%

Table 2. Match results. P.W. = progressive widening. Pat. = patterns in simulations.

- GNU Go 3.6
- Opteron 2.2 GHz:
  15,500 sim/sec (9×9), 3,700 sim/sec (19×19)

Conclusion / Future Work

- Generalized Bradley-Terry model is a powerful technique for pattern learnung
  - simple and efficient
  - allows large number of features
  - produces probability distribution over legal moves for MC
- Principle of Monte Carlo features could be exploited more
- Validity of the model could be tested and improved:
  - Use only one (or few) sample per game to improve independence of samples
  - Test linearity hypothesis of Bradley-Terry model (strength of team is sum of strength of members)
    Estimate the strength of some frequent feature pairs