2A3-5

囲碁における正確な着手予測のためのファジーパターンマッチング

Fuzzy Pattern Matching for Accurate Move Prediction in the Game of Go

荒木伸夫 Nobuo ARAKI　吉田和弘 Kazuhiro YOSHIDA　鶴岡慶雅 Yoshimasa TSURUOKA　辻井潤一 Jun'ichi TSUJII

東京大学情報理工学系研究科コンピュータ科学専攻
The Department of Computer Science, Graduate School of Information Science and Technology, The University of Tokyo

We address the problem of move prediction in the game of Go by performing machine learning with game records of expert players. Existing systems use straight-forward ways for representing stone positions in extracting features for machine learning. However, in Go, big patterns cause the data sparseness problem because there is not a sufficient number of game records for the complexity of the game. To tackle this problem, we propose novel methods for pattern extraction and pattern matching using a "fuzzy" pattern representation. This representation treats a mass of stones distant from the current move position fuzzily, as a compressed single stone. The compression of stone positions can alleviate the sparseness problem. To evaluate this representation, we carried out experiments with machine learning on expert games of the GoGoD database, but we couldn't improve the accuracy of move prediction by this representation.

1. はじめに

これまでに、囲碁プログラムに関する研究は数多くなされてきた。それにもかかわらず、囲碁プログラムは未だに弱く、アマチュア初段程度の棋力にとどまっている。チェス、将棋のプログラムと比べて、囲碁プログラムの棋力向上は非常に遅い。

囲碁プログラムが弱い理由の一つは、盤面が広いことである (囲碁:19 × 19、チェス:8 × 8、将棋:9 × 9)。盤面が広いため、選択肢が非常に多く、単純なサーチアルゴリズムでは十分な探索が出来ないのである。*1

この問題を解決する方法のひとつは、プロの棋譜を用いた機械学習による move prediction(着手予測) である。これは、囲碁の合法手を、プロが打ちそうな手が高位になるように、探索無しでランク付けすることである。この move prediction により、囲碁プログラムは探索を効率よく行うことが出来るようになる。$\alpha - \beta$ サーチの際のノード並べ替え [1] や、悪手の枝狩りに使えるからである。

move prediction に関する既存研究はいくつかある。そのうち現在最高精度のものは、Stern らによる研究 [2] である。この研究は、単純なパターンマッチにより、26% の精度 (テストデータとして 与えた盤面のうち 26% で、プロが実際に打った手を一位にランク付けした) を達成した。

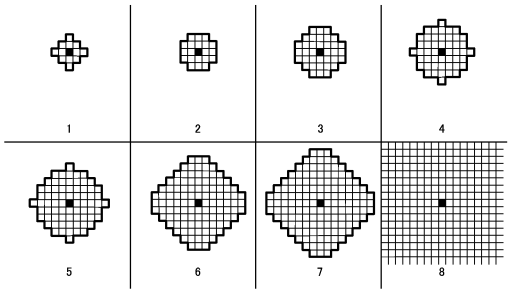
本研究では、「候補手から離れた石に関しては、ある範囲の石をまとめてひとつの石として扱う」パターンを提案する。これをファジーパターンを呼ぶことにする。このファジーパターンにより、sparseness の問題を解決して精度を上げようと試みた。

2. 基にした既存研究について

今回基にした Stern らの研究では、単純な石配置のパターンを素性として用いた機械学習により、move prediction を試みている。以下、この研究について説明する。

連絡先: 荒木伸夫, 東京大学情報理工学系研究科コンピュータ科学専攻辻井研究室, 電話:03-5803-1697,E-mail:ark@is.s.u-tokyo.ac.jp

*1 囲碁では、9 × 9 盤も使われるが、この盤の大きさでも、(19 × 19 よりはましだが) 囲碁プログラムはあまり強くない。それは、盤面の評価関数を作るのが難しいことによる。



各テンプレートの■は，そのテンプレートの中央を表す。パターン抽出の際、この中央が次の候補手の点に重なる。最大のテンプレート (8 番) は、盤全体をカバーする。

図 1: 単純パターンのテンプレート

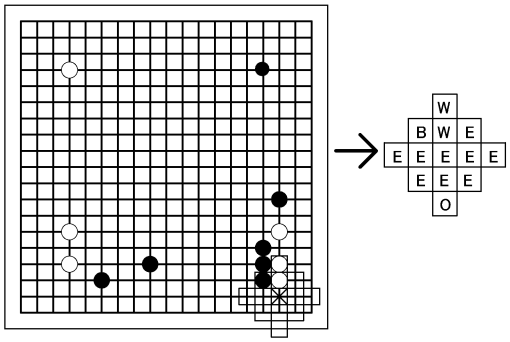
2.1 単純パターン

まず、単純パターンのテンプレートを準備する。Stern らの研究で用いられたものは、図 1 の 8 種類である。

これらのテンプレートを用いて、プロの棋譜から、単純パターンの抽出を行う。図 2 は単純パターンの抽出の一例である。図のように、単純パターンでは、各点は '黒'、'白'、'空き'、'盤外' の 4 通りの値をとる。

なお、回転対称、線対称、色反転、およびそれらを組み合わせた対称関係になっている単純パターンは、同一の単純パターンとして扱われる。

単純パターンの比較と記録を行う際は、単純パターンそのものを使うのではなく、それらのハッシュ値を使う。それにより、比較にかかる時間と、記録に要する容量が節約できる。ハッシュ値の計算方法としては、64bit の Zobrist hashing [3] を用いる。もとの局面のハッシュ値に基づいて、そこから進んだ局面のハッシュ値を計算できるという長所が、機械学習の際に重要な意味を持つ (詳細は 2.2 で述べる)。また、ハッシュ値を用いることにより、対称な単純パターンを同一のものとして扱うことが容易になる。ある単純パターンに対し、それと対称なパ



この図は、テンプレート (1 番) による、棋譜からの単純パターン抽出の例を示している。'B' は黒 (black)、'W' は白 (white)、'E' は空き (empty)、'O' は盤外 (out of board) をあらわす。

図 2: 単純パターンの抽出の一例

ターンのハッシュ値を全て計算し、その中から最小のものを選ぶようにすればよいのである。

2.2 統計手法

Stern らのシステムは、20,000 局のプロ棋譜 (GoGoD database [4], April 2003 より) による学習を行った。まず、その 20,000 局で、単純パターンの辞書を作り、そして、それらの単純パターンのスコア付けを、同じ 20,000 局で行った。

単純パターンの辞書作成は、訓練データの棋譜から、各テンプレートに対し、その中央が次のプロの着手点となるように単純パターンを取り出すことによって行われる。要するに、ここで言う辞書は、プロが (その中央を着点として) 選んだことのある (よって、選ぶ可能性のありそうな) パターンの集合である。ただし、大まかに見積もって、20,000 局の平均 250 手の棋譜に対して 8 個の異なるテンプレートにより単純パターン抽出を行うと、最高 40,000,000 種類の単純パターンが抽出されることになる。この数値が大きすぎるように思われることと、まだ見ぬ単純パターンに対する一般性のため、2 回以上出現したパターンだけが辞書に含まれるようにする。

次に、単純パターンのスコアを機械学習する。その学習ルーチンを以下に述べる。

各単純パターンのスコアは、BPM 分類機 (EP 使用) と ADF [5] により学習される。

まず、辞書中の各パターンのスコアを $p(s) = \mathcal{N}(s;\mu,\sigma^2)$ $(\mu = 0, \sigma = 1)$ の正規分布で初期化する。それから、各スコアは、棋譜上の各着手ごとに、ADF を用いて (各パターンに対して 1 次元の ADF) インクリメンタルにアップデートされる。その際、ADF に入力するデータポイント $\mathbf{x}_i$(スコアの候補値となる) は BPM 分類機 (EP 使用) によって決定される。すなわち、

プロ棋譜上の各着手ごとに、

1. 盤面上の全合法手それぞれについて、辞書から、その合法手を中央とし盤の現在の状態とマッチする、最大の単純パターンを抽出する。

2. 取り出されたパターンに対し、実際にプロが次の手として打った場所を中央とするパターンを正例、それ以外のパターンを負例として、BPM 分類機を学習させる。prior は現在のスコア、データポイントは以下のようなベクトル $\mathbf{x}_i$:

$$n \quad : \quad \text{合法手の個数} \tag{1}$$

$$\mathbf{x}_i \quad : \quad n-\text{次元ベクトル} \quad (i = 1, ..., n-1) \tag{2}$$

$$(\mathbf{x}_i)_j \quad = \quad \begin{cases} 1 & (j = e : \text{プロが実際に打った手}) \\ -1 & (j = a : \text{プロが選ばなかった合法手} \\ \qquad a = i & (i < e) \\ \qquad a = i+1 & (i \geq e)) \\ 0 & (\text{それ以外}) \end{cases} \tag{3}$$

これは、BPM 分類機が以下のように学習されることを意味する。

$$\begin{aligned} & (\text{中央がプロの着手のパターンスコア}) \\ \geq \quad & (\text{中央がその他の合法手のパターンスコア}) \end{aligned} \tag{4}$$

3. 結果として得られたベクトルの各要素を ADF データポイントとして使い，学習する。

このアルゴリズムにおいて、Zobrist hashing の長所 (もとの局面のハッシュ値に基づいて、そこから進んだ局面のハッシュ値を計算できる) が、ハッシュ値の計算時間節約に重要な役割を果たす。その長所を生かす以下のようなアルゴリズムにより、ハッシュ値は、毎回直接的な方法で計算するよりもはるかに速く計算できるのである。

1. 対局開始前の状態 (盤上の石が 0 又はハンデキャップ用の置石だけの状態) で、盤面の各交点に対し、各テンプレートについて、全対称パターンのハッシュ値を計算する。

2. 棋譜上の各着手について、その着手をテンプレート内に含むようなパターン全てのハッシュ値を、更新する。

2.3 Move prediction

Section 2.2 で得られたスコアを基に、move prediction を以下のように行う。Stern らの研究では、500 局 のテストデータを使ってこれを行い、26% の精度を出した。

1. 盤面上の全合法手それぞれについて、辞書から、その合法手を中央とし盤の現在の状態とマッチする、最大の単純パターンを抽出する。

2. 合法手を、抽出されたパターンのスコア (正規分布の期待値) によりランク付けする。

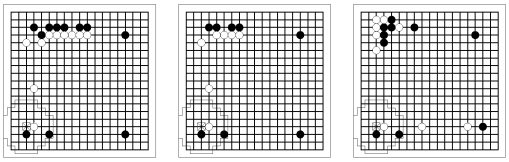
3. 提案手法:Fuzzy pattern

3.1 単純パターンの欠点

Section 2. で述べた単純パターンには、sparseness の問題がある。単純パターンが大きくなるほど、その単純パターンは石の配置に敏感になる。その結果、訓練データの量が不十分になり、大きな単純パターンのスコアをを正しく学習できない。

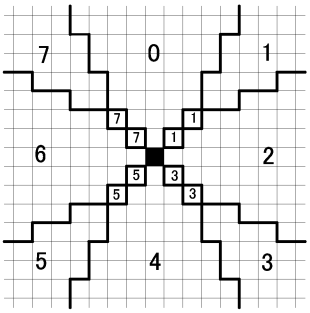
しかし、小さな単純パターンだけ使えば良いというわけでもない。図 3 は、小さな単純パターンがうまく作用しない例である。この図で、単純パターン内の石の配置は全て同じであるが、単純パターン外の石の配置は異なる。そして、パターンの中心に対する着手の価値も異なる (図 3 左と真ん中の ⊠ は良い手だが、右の ⊠ はあまり良くない。)。

単純パターンのみで、これらの違いを表すことは難しい。小さな単純パターンでは不十分なのは自明である。さらに、sparseness の問題から、大きなパターンを使うことにも問題がある (莫大な学習データが必要)。



次は白番。これらは異なる盤面の状態だが、図に示された単純パターン (テンプレートは 3 番) は全く同じである。

図 3: 小さな単純パターンの欠点



■はこのテンプレートの中心を表す。パターン抽出の際、この■が候補手に重なる (単純パターンテンプレートの場合と同様)。このテンプレートは盤全体をカバーする。

図 4: ファジーパターンテンプレート

3.2 提案手法:ファジーパターン

本研究では、ファジーパターンを提案する。ファジーパターンは、単純パターンを図 4 のファジーパターンテンプレートで拡張することによって作る。ファジーパターンは、単純パターンテンプレート 1 個と、ファジーパターンテンプレートを使って作られる。ファジーパターンを抽出する際は、単純パターンテンプレート内の石配置はそのまま取り出し、単純パターンテンプレート外かつファジーパターンテンプレート内の石配置は、「ファジーに」取り出す。「ファジーに」取り出すとことを数式で表すと、以下のようになる。

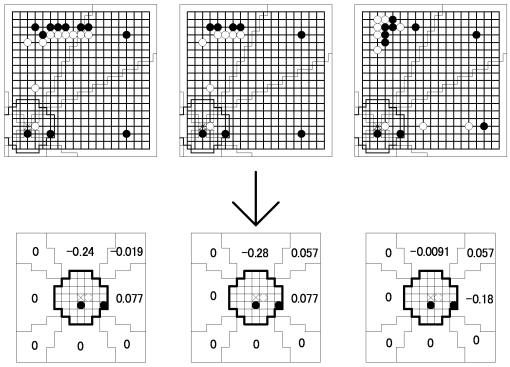
$$(\text{領域 } A_i\text{の値}) = \sum_{s\in A_i} (c(s)/(s \text{ までの距離})) \quad (5)$$

$$(i = 0, \ldots, 7)$$

$$c(s) = \begin{cases} 1 & (s : \text{黒石}) \\ -1 & (s : \text{白石}) \\ 0 & (s : \text{その他}) \end{cases} \quad (6)$$

図 5 は、ファジーパターン抽出の一例である。ファジーなパターン抽出は、勢力の大まかな抽出と言える。

どの色が、どの方向に対して、どの領域で勢力を張っているかを考えることは、碁の戦略において非常に重要である。ファジーパターンを用いれば、このような考えをパターンに取り入れることが出来る。さらに、ファジーパターンは sparseness の問題を緩和する。なぜなら、ファジーパターンは、小さな単純パターンにわずか 8 個のパラメータを加えただけで、盤全体をカバーするからである。



図中の数値は、「正の値→黒の勢力が強い」、「負の値→白の勢力が強い」ことを表す。閾値を設定することにより、これらの数値を黒、白、中間の三通りに分類できる。適切に閾値を設定すれば、左、中央と、右との違いを区別できる。

図 5: ファジーパターン抽出の例

また、単純パターンに対して用いた機械学習手法は、ファジーパターンに対しても通用する。閾値を設定して、実数を'黒'、'白'、'中間' に直せば、Zobrist hashing によるハッシュ値計算は出来る。しかも、式 (6) により、インクリメンタルなハッシュ値更新も、実数の加算減算を使うことで可能である。また、パターンの大きさの順序は、以下のように定義する。

$$\begin{aligned} t_i &= (i \text{ 番の単純パターンテンプレート}) \\ s_i &= (t_i\text{を使って作られた単純パターン}) \\ f_i &= (t_i\text{を使って作られたファジーパターン}) \\ \text{この時} &\quad s_{i-1} < s_i, \quad f_{i-1} < f_i, \quad f_{i-1} < s_i < f_i \end{aligned} \quad (7)$$

3.3 相対頻度による機械学習の試み

Stern らの研究では、BPM 分類機 (EP 使用) と ADF [5] を機械学習の手法として用いていた。しかし、この手法だと、学習に時間がかかる。学習時間短縮のため、本研究では、単純な相対頻度によるスコア付けも試みた。

アルゴリズムは以下の通りである。

1. 辞書中の各パターンに対し、二つの変数 p,n を用意する。
2. 訓練データの棋譜中の、プロの一手ごとに、
   (a) 盤面上の全合法手それぞれについて、辞書から、その合法手を中央とし盤の現在の状態とマッチする、最大のパターンを抽出する。
   (b) 抽出された各パターンに対し、その中央が実際のプロの次の手ならば、p をインクリメントする。そうでないならば、n をインクリメントする。
3. 各パターンのスコアを、以下のように計算する (ラプラススムージング [6] を用いている)。

$$\frac{p+1}{p+n+2} \quad (8)$$

表 1: 相対頻度による学習の結果

プロの手のランク	累積分布 (単純な相対頻度による実験)	累積分布 (Stern の論文中の値)
1	25%	26%
5	55%	55%
10	67%	68%
20	79%	81%

表 2: パターンの比較

プロの手のランク	累積分布 (単純)	累積分布 (ファジー)	累積分布 (混合)
1	21.0%	12.3%	19.7%
2	30.8%	17.2%	29.0%
3	37.2%	20.0%	35.3%
4	42.1%	22.0%	40.1%
5	46.2%	23.4%	44.2%

4. 実験

4.1 設定

我々は、GoGoD database [4](December 2005) から、ランダム抽出により、以下の 4 個の棋譜セット (排他的) を作った。括弧内は棋譜の数を表す。

- Training(20,000),Testing(500) ([データ 1])
- Minitraining(2,000),Development(500) ([データ 2])

[データ 1] は、Stern らの実験との比較に用いる。[データ 2] は、ミニ実験用である。

ファジーパターンの閾値は、以下のように設定した。

$$
\begin{aligned}
M &= \max_{i=0\ldots7}(\text{領域 } A_i\text{の値}) \qquad (9)\\
m &= \min_{i=0\ldots7}(\text{領域 } A_i\text{の値})\\
(A_j\text{に対応する色}) &= \begin{cases} \text{黒} & (A_j > \frac{2}{3}M + \frac{1}{3}m)\\ \text{白} & (A_j < \frac{1}{3}M + \frac{2}{3}m)\\ \text{中間} & (\text{上記以外}) \end{cases}
\end{aligned}
$$

4.2 統計手法の比較

Stern らが用いたのと同じ単純パターンを使って、単純な相対頻度による学習を [データ 1] に対して行ったところ、表 1 のような結果になった。 この結果は、単純な相対頻度による学習でも、Stern らの論文 [2] 中の数値に近い精度が出ることを示している。

しかも、相対頻度による学習は、Stern らが用いた BPM(EP 使用) と ADF による学習と比べて、はるかに速い。我々の実装では、9〜10 倍の速度差があった。

よって、これ以降の実験は全て、単純な相対頻度による学習で行う。

4.3 パターンの比較

[データ 2] を用いて、8 種類の単純パターン (図 1 の 8 種類のテンプレートから作られる) による学習と、8 種類のファジーパターン (図 1 の 8 種類のテンプレートと、図 4 から作られる) による学習と、単純パターン+ファジーパターン混合 (単純テンプレート 8 番から作られるパターンは双方一致するので、全部で 15 種類) による学習との結果を比較する。

結果は表 2 の通りであり、ファジーパターンはうまく働いていないように思われる。

5. 結論

先の結果から分かるように、今回の実験では、ファジーパターンによって精度を向上させることは出来なかった。むしろ悪化してしまった。

しかし、ファジーパターンが悪いと結論付けることは出来ない。アルゴリズムを改良することで、ファジーパターンがうまく働くようになるかもしれない。今回実験したものは、あくまでファジーパターンとして考えられるものの一例である。閾値の決め方を変える、ファジーパターンテンプレートを変える、ファジーパターンでの値の計算方法を変える等で、精度が上がるかもしれない。これらのいろいろな組み合わせを試すことが、今後の課題である。

参考文献

[1] E. van der Werf. *"AI techniques for the game of Go."*. PhD thesis, Universiteit Maastricht, 2004.

[2] D. Stern, R. Herbrich, and T. Graepel. "Bayesian Pattern Ranking for Move Prediction in the Game of Go.", 2005. Draft.

[3] A. Zobrist. "A new hashing method with applications for game playing.". *ICCA Journal*, (13(2)), 1990.

[4] T. Mark and J. Fairbairn. GoGoD. [Online; accessed 6-February-2006].

[5] T. P. Minka. *"A family of algorithms for approximate Bayesian inference."*. PhD thesis, Massachusetts Institute of Technology, 2001.

[6] 北研二. 言語と計算-4 確率的言語モデル, 1999.

[7] E. van der Werf, J. Uiterwijk, E. Postma, and J. van den Herik. "Local move prediction in Go.". 2002. In 3rd International Conference on Computers and Games, Edmonton.

[8] B. Bouzy and G. Chaslot. "Bayesian generation and integration of K-nearest-neighbor patterns for 19x19 go". IEEE 2005 symposium on computational Intelligence in Games, Colchester, UK, G. Kendall & Simon Lucas (eds).

[9] Opper M. and Winther O. "Gaussian Processes and SVM: Mean field results and leave-one-out.". In *Advances in Large Margin Classifiers*. MIT Press, 2000.

[10] Csato L., Fokue E., Opper M., Schottky B., and Winther O. "Efficient approaches to Gaussian process classification.". *NIPS*, (12), 1999.

[11] F. de Groot. Moyo Go Studio. [Online; accessed 6-February-2006].