SVMによる将棋の詰みの予測とその応用

三輪　誠[†]　横山大作[†]　近山　隆[†]

将棋において、詰みの有無を判断することは直接勝敗につながるため非常に重要な問題である。本研究では、この詰みの有無の判定問題に対する1つのアプローチとして、SVMを用いた将棋の詰み有無の予測の学習を行った。今回は40000局面のサンプルを用いてこの方法を試し、探索内で呼ばれる詰探索の探索ノード数制限を制御することを試みた。結果として、86%の正解率、平均10μsec以下で分類できる非常にコストの低い分類器が得られ、その分類結果に応じて詰探索の探索ノード数制限を制御することで探索の精度を損じることなく探索時間を62%まで削減できた。

Prediction of mates in shogi using SVM and its application

MAKOTO MIWA,[†] DAISAKU YOKOYAMA[†]
and TAKASHI CHIKAYAMA[†]

Checking mates is a critical problem in shogi. In this paper, we introduce a new way to check mate, learning the probability of mate in shogi using support vector machine. We tested this way using 40000 positions, and tried to control the upper bound of search nodes of mate searches called within search. The experimental results show this way is useful because of about 86% rate in accuracy, the fast classification enough for practical use, and the control reduced the search time by about 62% without losing the accuracy of search.

1. はじめに

機械学習を用いて、強いゲームプレイヤを作るという研究[11]は1950年代から行われている。特にチェッカーやバックギャモン、オセロなど比較的簡単なゲームにおいては人間のプロに勝つ、世界のトップレベルのコンピュータゲームプレーヤとほぼ同等の実力を持つなど、非常に良い成績を挙げている。また、チェスや将棋、囲碁[2]、Amazonなど比較的難しいゲームにおいても、ニューラルネットワークやTD法[1]を用いて評価関数の一部を最適化する、などゲームの一部を最適化する研究が盛んに行われており、好成績とは言えないまでも、それなりに良い結果を出している。

このような「ゲームの一部」に含まれるものとして、将棋の詰み判定が挙げられる。将棋の終盤において、詰みの有無を判断することは直接勝敗につながるため、非常に重要な要素である。そのため現在の将棋プログラムでは、特に終盤において通常の探索の中で独自の詰みチェックアルゴリズムを呼び出し、それに多くの時間をかけることが一般的になっている。例えば、東京大学近山研究室の将棋プログラム『激指』では、全探索時間の30%の時間を詰みチェックに用いている。この詰みチェックアルゴリズムには、PDS、df-pnなど証明数、反証数を用いた詰将棋のための探索アルゴリズムが用いられることが多い。これらのアルゴリズムが非常に高速であり有用であることは、現在の将棋プログラムが詰みの発見という部分においてはプロを上回っているといわれていることからも明らかである。しかし、その一方で詰みがないことの発見に関しては、詰みの発見に比べてその探索に多くの探索ノード数を必要とすることがわかっている。そのために、探索量を多く割り当てることが必要となるが、それでも結果がわからないときには不明という結果しか得られず、多くの探索時間をほとんど意味のない探索に費やしてしまうことにもなりかねない。このような点はこれらのアルゴリズムを利用する際、問題となる部分であり、改善の余地がある。

このような詰みの有無の判定問題に対する1つのアプローチとして、本研究ではSupport Vector Machine (SVM)を用いた将棋の詰み有無の予測の学習を行った。また、その分類器を用いた応用例として、詰探索PDSの探索ノード数制限をその分類器の結果に応じて変化させることを試みた。

† 東京大学大学院新領域創成科学研究科
Graduate School of Frontier Sciences, University of Tokyo

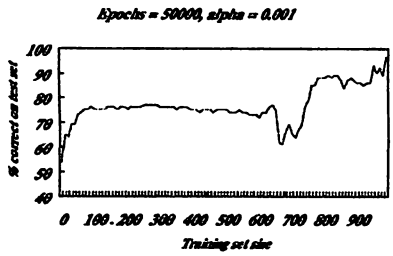

図 1 Learning Curves for the perceptron. (Grimbergen, 2002)

結果としては、実用に耐える計算時間で分類ができ、86.0%というそれなりの高い正解率を得ることができた。また、PDS の探索ノード数制限の制御については、同等かそれ以上の強さを保ちながら探索時間を最大で 61.9%まで削減できるという結果が得られた。

これ以降の本論文の構成は次のようになっている。まず 2 で関連研究について述べる。次に、3 で今回用いた Support Vector Machine と本研究の手法について説明する。4、5 で本研究の実験方法とその結果をそれぞれ述べた後、6 で本研究全体について考察し、7 で本論文のまとめと今後の課題について述べる。

2. 関連研究

この章では、機械学習の将棋への適用例、詰探索アルゴリズム PDS について紹介する。

2.1 機械学習の将棋への適用

機械学習を用いてコンピュータ将棋プレイヤを強くする研究はパターン認識[5)7)] を用いたもの、ニューラルネットワークを用いたもの、TD 法[14)] を用いたもの、統計情報[16)17)13)] を元にしたもの、など多くの研究がなされている。ここでは、SVM に性質の近いニューラルネットワークを用いたものを紹介する。

Grimbergen ら[6)] は単層パーセプトロンを用いて、王の危険度の評価を行った。入力には 161 要素の特徴ベクトルを用い、500 個の中盤の局面を正例、500 個の詰将棋を負例として、そのデータを訓練データとテストデータにわけ、学習率 0.01 と 0.001 それぞれについて、学習回数を 5000 回から 50000 回まで 5000 ずつ変化させて実験を行っている。結果は図 1 のようになっており、訓練データが多いときは 90%以上の正解率を示している。

2.2 詰探索アルゴリズム

詰探索の研究は、詰将棋に関する研究がほとんどである。詰将棋は 1970 年代から研究がなされているが、特に 1990 年代初め AND/OR 木探索の問題として考えるようになってから、急速に発展してきた。特に証明数、反証数を用いた探索手法は、1997 年に 1,525 手詰のミクロコスモスを解くことに成功するなど、非常に良い結果[12)] を示している。このような探索アルゴリズムには PDS(Proof-number and Disproof-number search)[8)] や df-pn(depth first proof number search)[9)] があげられる。ここでは、本研究の評価の 1 つに用いた PDS について紹介したのち、指将棋プログラム内でのその利用について述べる。

2.2.1 PDS

PDS[8)] とは、長井によって 1998 年に提案された深さ優先の探索手法である。PDS は、できるだけ少ないノードの展開でそのルートノードが詰み・不詰であることを証明しようとする手法である。そのために PDS は、ルートノードが詰み・不詰であることを証明するために必要な展開ノード数が、少ないほうを優先して探索を行っていく。

PDS の特徴としては次のようなものが挙げられる。

- 証明数、反証数の両方を用いる。
- 探索結果を覚えておく Transposition Table を用いて、ルートノードから多重反復深化 (multiple iterative deepening) を行うことで、(証明数、反証数が少ないということを最良とした) 最良優先探索と同様の効率的な探索が比較的少ないメモリ量でできる。

ここで、証明数、反証数とは次のように定義される数である。まず、証明数 (proof-number, pn) とは、ルートノードの評価値が true(将棋では詰み) になるには、最低あといくつの葉ノードの評価値が true である必要があるか、を示す値であり、次のように定義される。

Definition 1 証明数

(1) n が葉ノードのとき

(a) 評価値が *true* のとき $n.pn = 0$

(b) 評価値が *false* のとき $n.pn = +\infty$

(c) 評価値が不明のとき $n.pn = 1$

(2) n が内部ノードのとき

(a) *OR* ノードのとき

$$n.pn = \min_{n_c \in children\ of\ n} n_c.pn$$

(b) *AND* ノードのとき

$$n.pn = \sum_{n_c \in children\ of\ n} n_c.pn$$

次に、反証数 (disproof-number, dn) とは、ルートノードの評価値が false(将棋では不詰) になるには、最低あといくつの葉ノードの評価値が false である必要があるか、を示す値であり、次のように定義される。

Definition 2 反証数

(1) n が葉ノードのとき

(a) 評価値が *true* のとき $n.dn = +\infty$

(b) 評価値が *false* のとき $n.dn = 0$

(c) 評価値が不明のとき $n.dn = 1$

(2) n が内部ノードのとき

(a) *OR* ノードのとき

$$n.dn = \sum_{n_c \in children\ of\ n} n_c.dn$$

(b) *AND* ノードのとき

$$n.dn = \min_{n_c \in children\ of\ n} n_c.dn$$

PDS は上記のように定義される証明数、反証数を用いて次のように探索する。まず、証明数、反証数の閾値を (1, 1) に設定する。そして、ルートノードから証明数もしくは反証数の小さいノードから、ノードの閾値をルートの閾値を越えないように徐々に増やしながら順に展開する。探索中、証明数、反証数が閾値を超えたノードについてはそこで展開を打ち切る。このような探索を行い、ルートノードの証明数、反証数が閾値を超えた場合は、証明数、反証数いずれかの閾値を増やしてルートノードから再び探索を行う。このような探索を繰り返し、ルートノードの証明数もしくは反証数が 0 になった場合、それぞれルートノードが true, false であるとわかるので、探索終了となる。

このアルゴリズムを将棋プログラムに用いる際には、一定時間に探索が終わらないことが考えられる。そのため、将棋プログラムに用いる際には、展開ノード数に閾値を設定し、閾値を越えてもルートノードの値が一意に決まらない場合は不明として探索を終了する。

2.2.2 指将棋プログラム内での詰探索アルゴリズムの利用

指将棋プログラム内では、一般にいわゆる「必至」のような連続王手でない詰みや、こちらが何もしなければ詰まされてしまう「詰めろ」をみつけるために探索中のノードほとんどすべてにおいて詰探索を行う。この詰探索は浅いところでは多く、深いところでは少なくノード数を制限して探索を行い、深すぎるところでは時間の都合上呼ばれないことが多い。

3. SVM を用いた将棋の詰み有無の予測の学習

本章ではサポートベクタマシンについて説明をした後、本研究の手法である SVM を用いた将棋の詰み有無の予測の学習について説明する。

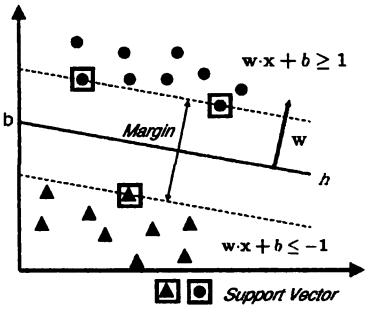

図 2 Support Vector Machine

3.1 Support Vector Machine

Support Vector Machine(SVM)[4] は、1992 年に Vaptik らによって提案された 2 クラスの分類器の 1 つである。

SVM は、超平面とそれに最も近い例との距離 (マージン) を最大化するように、訓練データを線形分離する (分離) 超平面を求める。その特徴として主に次の 3 つが挙げられる。

- 少ないサポートベクタで分割境界の超平面を張ることで、計算量を少なくできる。
- 非線形領域へのカーネル関数を用いたマッピング (カーネルトリック) により、非線形の分割問題を計算量をあまり増やさずに分割可能な問題に変換できる。
- 出力はシグモイド関数に当てはめることで、確率としても利用可能である。[10]

今回は、主に SVM の線形分離問題への適応についてその超平面を求める方法を説明する。

図 2 のように、$\mathbf{w}$ を超平面の法線ベクトルとし、$\mathbf{x}_i$ が正例ならば+1、負例なら-1 となるものを y_i とする。

この図は 2 次元であるが、SVM ではこのように 2 クラスに分かれた集合の (分離) 超平面 h を動かして、超平面とそれに最も近い例との距離 (マージン) が最大になるようにして分類器を構築する。

このとき、マージンの長さは $2/||\mathbf{w}||$ となるので、マージンを最大化するということは $||\mathbf{w}||$ を最小化すればよい。そのため、問題が線形分離可能であれば、以下のように表現できる。

$$\text{Minimize } ||\mathbf{w}||$$
$$\text{subject to } y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1 \qquad (1)$$

この問題はラグランジュ未定乗数を導入することにより以下の双対問題に変形できる。

$$\text{Minimize } -\sum_{i=1}^{n}\alpha_i + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j$$

$$\text{subject to } \sum_{i=1}^{n}\alpha_i y_i = 0 \text{ and } \forall_i \alpha_i \geq 0 \qquad (2)$$

このとき Karush-Kuhn-Tucker 条件

$$\alpha_i(y_i(\mathbf{w}\cdot\mathbf{x}_i + b) - 1) = 0 \qquad (3)$$

が要求されるため、超平面から一番近いデータの α 以外は全て 0 になる。これらの超平面から一番近いデータをサポートベクタ (Support Vector, SV) と呼ぶ。

この式 (2) の最適解 α^* を用いて

$$\mathbf{w}^* = \sum_{\mathbf{x}_i \in SV} \alpha_i^* y_i \mathbf{x}_i \qquad (4)$$

$$b = y_k - \mathbf{w}^* \cdot \mathbf{x}_k,\ \mathbf{x}_k \in SV \qquad (5)$$

と求めることができる。これにより、分類器は

$$f(\mathbf{x}) = sgn(\mathbf{w}^* \cdot \mathbf{x} + b) \qquad (6)$$

$$= sgn\left(\sum_{\mathbf{x}_i \in SV} \alpha_i^* y_i (\mathbf{x}_i \cdot \mathbf{x}) + b\right) \qquad (7)$$

と求めることができる。このように分類器は SV のみを用いて計算すればよく、また一般に SV は入力データの数に対して非常に少なくなることが多いため計算量を少なくすることができる。また、(線形においては) 事前に $\mathbf{w}^*$ を計算しておくことにより、ベクトルの掛け算のみで分類ができ、非常に少ない計算量で分類が可能となる。

ところで、これまで、マージンを最大化すればよいということを前提に話を進めてきた。しかし、なぜそれでうまく行くのかについては何も触れていなかったので、ここで簡単にだが説明しておく。これは、Vapnik Chervonekis(VC) dimension を用いて説明することができる。VC dimension は、モデルの記述能力・複雑さを表し、それを d とすると

$$P_{true\ error} \leq P_{train\ error} + \sqrt{\frac{d(log(2l/d)+1) - log(\lambda/4)}{l}} \qquad (8)$$

となることが知られている。この式 (8) より汎化誤差を小さくするには d を小さくすればよいことがわかる。また、SVM では d は、すべての入力ベクトルを含む最小の球の半径を R、入力ベクトルの数を n とすると、

$$d \leq min(R^2||\mathbf{w}||^2,\ n) + 1 \qquad (9)$$

と制限されることが知られている。この式 (8), (9) より $||\mathbf{w}||$ が小さければ汎化誤差が小さくなるということになる。このことからマージンを最大化すればよいということになる。また、SVM は学習エラー (式 (8) の右辺第一項) を固定しながら、式 (8) の右辺第二項を最小化するという方法をとる。これにより SVM は過学習しにくい (汎化能力に優れた) 分類器となっている。

3.2 SVM を用いた将棋の詰み有無の予測の学習

本研究では、SVM を用いて将棋の詰みの有無の予測を学習する。ここで予測といっているのは SVM で 100%の詰み判定ができないことは明らかであり、3.1 にも示したとおり、SVM における出力はシグモイド関数に当てはめることで詰みの確率としてみることができることがわかっているので、それを使ったほうが有用ではないか、と考えたためである。

学習の方法は次のようになる。

(1) 学習する局面に詰み・不詰のラベルをつける。
(2) 学習する局面から特徴量を取り出し、ラベルを元に正例、負例として SVM に入力する。

ここで、どのような局面を詰みと判断するか、特徴量はどのような量を取り出すのかという問題があるが、これは 4.1 で説明する。

この分類器が有用であるかどうかを判断するには次のような多くの判断基準が考えられる。

(1) 正解率が高いか。(うまく分類できているか。)
(2) 分類器を PDS などの詰探索アルゴリズムの探索順序制御 (move ordering) に適用することで探索ノード数を減らすことができるか。
(3) 通常の探索と詰みチェックアルゴリズムの計算時間の分配を調整した場合に従来のプログラムよりも強いか。
(4) 他の詰探索アルゴリズムの代用として利用できるか。
(5) 高速に計算できるか。
(6) 詰探索以外の場所でも利用できるか。

(2) と (4) については他の手法と同等かそれより悪い結果となることがわかっている[15]。本論文では、(1), (3), (5) の評価を目的として実験を行った。

4. 分類器の性能

本章では分類器の性能を計測するための実験方法とその結果について述べる。

4.1 実験方法

SVM を用いて将棋の詰み有無の予測の学習を行うにあたり、次のような設定で実験を行った。

4.1.1 詰みについて

将棋において、詰みは狭義では次のように定義することができる。

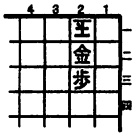

図 3　狭義の詰み

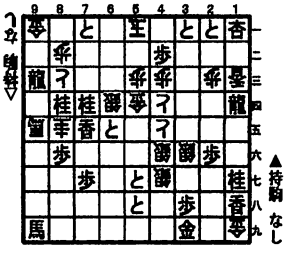

図 4　広義の詰み (ミクロコスモス)

Definition 3　狭義の詰み
どんな合法の手を指しても、次の相手の手において玉が相手に取られる局面。

図 3 に狭義の詰みの一例を示す。この局面では、玉がどこに動いても金、もしくは歩に取られてしまうため詰みである。

また、広義において詰みは次のように定義することができる。

Definition 4　広義の詰み
連続王手で正しい手順を指す (「詰ます」) ことで、狭義の詰みに持ち込める局面。

この定義では、詰将棋全般は詰みであるということができる。図 4 に広義の詰みの例を示した。

このように詰みを定義することができるが、狭義の詰みは王が動けるか否かの判定を直接将棋盤を見て判断するだけでよいため、詰み判断に用いる定義としては狭すぎる。また、その一方で広義の定義は詰み判断に用いる定義としては広すぎる。これは、図 4 のような難解な問題は詰みと判断できてもコンピュータ将棋の一般的な制限時間である 30 秒以内に探索するのは (少なくとも現在の環境では) 不可能なためである。

このようにどちらの定義も実際のコンピュータ将棋の詰み判断に用いるには適切でない定義であるため、本研究では次のように詰みを定義し、この定義に合わないものは不詰とした。

Definition 5　詰み
(近山・田浦研究室の将棋プログラム『激指』実装の)PDS で 5,000,000 ノード探索するうちに広義の詰みであると判定された局面。

ここで、5,000,000 ノードとは、現在の実験環境下で約 30 秒強で探索が終わる値である。より正確な詰む詰まないの情報を得るためにこれ以上のノード数を制限に用いることも考えられるが、

- 詰む詰まないを正確に判断することは現実的に不可能であること
- ゲーム中には PDS で探索できず不明と判断され SVM の判断を確かめることができないため不詰と考えても差し支えないこと

の 2 つの理由からこのような値を用いた。

4.1.2　特徴量について

SVM に入れる特徴量としては、将棋盤そのものを入力ベクトルとして入れて、カーネルトリック[4] などにより高次元に射影して分類することも考えられるが、単純な射影では分類がまったくできない、高次元への射影にコストがかかる、などの問題がある。そのため、ここでは将棋の知識を元に特徴量ベクトルを作ることにした。

SVM への入力としての特徴量には Grimbergen ら[6] の用いた 161 要素の特徴量を元に、その特徴量から追加、削除、統合した 129 要素のベクトルを用いた。具体的には次の特徴量を攻め手、守り手両方について取り、あわせたものを特徴量ベクトルとした。

- 王について
  - 王の絶対位置 (行、列)
  - 両玉のマンハッタン距離
- 王の周り 8 枡、王の前 14 枡、後 10 枡について
  - 端
  - 駒がある
  - 敵に有利または自分に有利 (利きの数がどちらが多いほうを有利とした)
  - 王の逃げ道の数
  - 守り駒 (王側の駒)、攻め駒 (敵側の駒)
  - ピンされている駒
- 持ち駒について
  - 持ち駒の総数
  - それぞれの駒の数
- 王への攻撃の可能性について
  - 王への利き (間接利き含む) ができる可能性のある指し駒の数と動ける位置の数
  - 王への利き (間接利き含む) ができる可能性のある持ち駒の置ける位置の数

王の周りについて前後の特徴量を分割したのは、将棋は左右には対称であるが、上下には非対称であるためである。分割した場合は、分割しない場合、もしくはすべての枡を統合しない場合に比べて正解率がそれぞれ 3%、1%程度改善することも事前実験において確

認した。また[18]のような局面の進行度を考慮した要素を取り入れてもほとんど効果が無いことも事前実験において確認した。また、今回挙ているこの129要素の中にも影響が大きいもの、小さいものがあり、小さいものを取り除くことで速度を改善させることも考えられるが、今回は行っていない。このように、特徴量としてどの特徴をとるかということは非常に重要な問題であり、上記の特徴量にも改善の余地がある。

4.1.3 データについて

データは次のように作成した40000局面を用いた。

(1) プロの対局4000局それぞれについて終局までの最後の10局面を取り出す。

(2) 取り出した40000局面それぞれについて浅く探索し、探索中に出てきたノードをランダムに1つずつ取り出す。

最後の10局面を取り出したのは、そこが詰み判定が最も有用とされる場面であり、この範囲で学習したものは、実際の詰み判定でも有用であると考えられるためである。また、それぞれに関して浅い探索を行ったのは、探索中には頻繁に出てくるが、対局中にはあまり出てくることのないような(機械的な)局面も分類器に判別させたいと考えたためである。

上記のようにして取り出した40000局面の中には詰みの局面が10159個あった。この詰みの割合は、終局に近いところ以外では、実際の将棋の傾向よりも大きい。そのため実際に利用する際には詰みと認識してしまう可能性が大きくなることを考慮する必要がある。このような問題が考えられるにもかかわらず、今回のような詰みに偏ったデータを用いたのには次のような理由がある。事前実験において、対局全体からランダムに40000局面取り出して(詰みの局面が120局面で)学習させたところ、このデータで学習したSVMに対してどちらのデータに関しても正解率が低いという結果が得られた。このような結果になったのは120局面という数が将棋の詰みの学習の要素数としては少なく、詰みの知識をほとんど得られなかったためだと考えられる。ここで、学習するデータ数を増やすことが考えられるが、将棋の詰みの学習に必要な局面数は非常に大きいことは明らかで時間の制約上現在の環境では困難である。このようなことから詰みの要素を増やすことが必要となり、今回のようなデータを用いた。

4.1.4 SVMについて

SVMは、SVMのライブラリであるLIBSVM[3]を線形専用に改良したものを使用した。

4.1.5 実験環境

実験はCPU Xeon 2.40GHz dual, メモリ2GBの環境で行った。

表1 分類器の性能

	詰み	不詰	合計
詰みと分類された数	6414	1865	8279
不詰と分類された数	3745	27976	31721
合計	10159	29841	40000

4.2 実験結果

分類器の性能については表1のような結果が得られた。この結果は4.1.3に示した40000局面のデータを4-fold cross validationで評価した結果である。この表1より正解率86.0%, Precision 0.775, Recall 0.631, f1 0.696という結果が得られた。また、計算時間を示しておくと、(ラベル付の)30000局面の学習に3時間近くかかったのに対し、分類には1局面あたり6.99μsec☆で終わるという結果がえられた。

この正解率は、学習データに関してfalse positiveはあるものの PDSで244ノード探索した場合と比較したときの値とほぼ同じ結果となっている。244ノードは『激指』で実際に探索する際、葉ノードから7段目を探索するときに使う制限であり、深いところでは『激指』の詰探索の代用に使うことも考えられる。また、PDSが142ノード/msec程度であることから、この分類器はPDSで約1ノード探索する時間で分類できており、この分類器は実用にも十分に耐える計算時間で分類ができているということがわかる。

5. PDSの探索ノード数制限の制御

この章では、分類器の応用の1つとして、分類器による結果を元に詰探索の探索ノード数制限を制御する手法について述べ、その実験結果を示す。

5.1 PDSの探索ノード数制限の制御

2.2.2に示したとおり、指将棋プログラム内では探索中の全てのノードで探索ノード数を制限しながら詰探索を行う。本研究では、この詰探索の探索ノード数の制限を分類器による結果に応じて制御した。

4.1.3の40000局面から学習して得られた分類器には不詰であると判断した場合に

- 90%以上の正解率で(特に探索に用いられるような数千ノード程度の少ないノード数制限で探索した結果との比較では98%以上の正解率で)不詰もしくは不明である。
- 探索しても不明という結果になることが多い。

という特徴があった。これらのことから、探索ノード

☆ 特徴量の一部は先読みに当たっての局面の更新時に差分計算をしているため、特徴量を新たに算出する場合よりは若干早くなっている。

表 2　制御をしない『激指』と 1000 回対戦した時の勝率と探索時間

	線形	シグモイド	階段
勝率 (n-sigma)	0.545 (2.77)	0.534 (2.11)	0.503 (0.16)
ゲーム全体の探索時間の比 (オリジナルを 1)	0.762	0.656	0.833

表 3　シグモイドにおける定数を変化させた場合の勝率と探索時間の変化

	1/2	3/4	1	2	4
勝率 (n-sigma)	0.448 (-3.23)	0.503 (0.16)	0.534 (2.11)	0.553 (3.26)	0.528 (1.73)
探索時間の比	0.907	0.619	0.656	0.768	0.900

数制限の変化の方法には、不詰と判断された場合は探索ノード数の上限を減らすという方法をとった。

また、不詰と判断された場合の探索時間を減らした分、詰みと判断されたときはその探索ノード数の上限を増やすことにした。これは

- 今回得られた分類器が詰みであると判断した場合に 78.2%が詰みであること
- 詰みを早めに見つけることでより正確な探索ができると考えられること

からこのようにした。また、この制御を行う際、分離超平面からの距離が詰む詰まないの予測の精度と相関が高いと考えられ、これをパラメータとする関数を用いることで、PDS の探索ノード数制限の制御を試みた。この関数として、次の 3 つの関数を用いた。

- 分離超平面からの距離にあわせて線形に変化する関数
- 分離超平面からの距離を基にしたシグモイド関数
- 分離超平面を境界に 2 値をとる階段関数

5.2 実験方法

5.1 に示した 3 つの関数を用いて制御を行った。評価は探索深さを固定して、制御したものと制御を行わない『激指』を 1000 局対局させることで評価を行った。

5.3 実験結果

探索全体の詰探索時間の比率が変わらないようにして対戦を行った場合の結果を表 2 に示した。ここで、n-sigma というのは勝敗を 2 項分布と仮定したときのシグマ値である。線形の関数については学習データの中で正例の中で分離超平面からの距離が最も大きいものが等倍、分離超平面が 1/2 倍になるようにして決定した。また、探索全体の詰探索時間の比率が変わらないようにするために関数全体を定数倍した。この結果を見ると、いずれの関数についても探索精度を変えずに、ゲーム全体の探索時間を大きく削減していることがわかる。この最も削減されたシグモイド関数を用いた手法について、対局をその手数について単純に 3 分割して初盤、中盤、終盤と分けた場合それぞれについてのゲーム全体の探索時間の比は 1.01, 0.635, 0.572 となった。このことから、この手法は詰みチェックが有用な中盤、終盤において特に有効だということがわかる。また、統計的に有意といえないまでも、全ての結果において勝率が 5 割を超えている。このような結果になったのは詰探索が正確になったためであると考えられる。

次にこのゲーム全体の探索時間が最も削減されたシグモイド関数を用いた手法についてその関数にかける定数の大きさ (探索全体の詰探索時間の比率が変わらないものを 1) を変化させた場合の勝率と探索時間の変化を表 3 に示した。この結果を見ると、勝率は 2 倍のときに最も大きくなっており、この時の勝率は統計的に有意であるということができる。また、3/4 倍のときその探索全体の探索時間の削減量は最大になっておりその強さを変えずに 61.9%まで削減、つまり約 4 割の時間を削減することができている。1/2 倍のときに、勝率が下がって、他の傾向と異なり探索時間が増えているが、これは PDS の探索ノード数制限がオリジナルよりも常に小さい値になっており、詰探索が不正確になったためであると考えられる。

6. 考　　察

今回の実験ではカーネルトリックなどを用いず線形の SVM を用いた。この線形でそれなりの分離ができているのは、将棋盤を特徴量に射影する際に線形に分離できる形に射影できているためであると考えられる。高速に分類するためには線形の SVM のような軽い計算ですますことは重要であり、今回用いたような特徴量を将棋盤から抽出することは非常に重要な問題である。また、今回用いたのは SVM であるが、このような軽い計算で分類できる学習器は他にもあり、それらも SVM と同様に有用であろうと考えられる。

また、今回の実験により探索時間が大量に削減できたことから、詰探索の制限をその深さに応じてのみではなく、その他の情報に応じてより柔軟に変化させる必要があるということがわかる。今回の実験で得られたものに限らずこのような詰探索を制御する情報はこれからのコンピュータ将棋をより強いものにするために必須であると考えられる。

7. おわりに

この章では本論文のまとめを行った後、今後の課題について述べる。

7.1 まとめ

本論文では、SVMを用いた将棋の詰み有無の予測の学習手法とその分類器の性能、その応用の1つとしてPDSの探索ノード数制限の制御への適用結果について紹介した。

結果としては、SVMを用いた詰みの有無の学習では86.0%というそれなりの高い正解率を持ち、ほとんどコスト無しに分類可能な分類器を得ることができた。特に探索ノード数が少ないところでの不詰判定器としては非常に高い精度で判別できることもわかった。また、PDSの探索ノード数制限の制御については、同等かそれ以上の強さを保ちながら、詰探索が有用な中盤、終盤にかけて、最大で探索時間を61.9%まで削減できることがわかった。

7.2 今後の課題

今後の課題としては次の3つがあげられる。

- 特徴量の選択
  将棋盤から取り出す特徴量の調整は、正解率に直接影響するため非常に重要な要素である。現在は人手で抽出しているが、単純な特徴量の組み合わせなどにより、これを自動化する試みも考えられる。
- より適切な関数の選択・制御パラメータの調整
  表2, 3を見てもわかるように関数とそのパラメータを変えることで勝率・探索時間は大幅に変化することがわかっており、より適切なものを見つけることが必要だと考えられる。
- 分類器の他の応用
  今回得られた分類器は、今回の応用のほかにも例えば候補手の探索順序制御や評価関数に組み込むなどの応用が考えられる。

参考文献

1) Baxter, J., Tridgell, A. and Weaver, L.: Learning to Play Chess Using Temporal Differences, *Machine Learning*, Vol. 40, No. 3, pp. 243–263 (2000).
2) Bouzy, B. and Cazenave, T.: Computer Go: An AI oriented survey, *Artificial Intelligence*, Vol. 132, No. 1, pp. 39–103 (2001).
3) Chang, C.-C. and Lin, C.-J.: LIBSVM: a Library for Support Vector Machines (2003). Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`.
4) Cortes, C. and Vapnik, V.: Support-Vector Networks, *Machine Learning*, Vol. 20, No. 3, pp. 273–297 (1995).
5) Grimbergen, R.: Using pattern recognition and selective deepening to solve tsume shogi, *Game Programming Workshop in Japan '96*, Kanagawa, Japan, pp. 150–159 (1996).
6) Grimbergen, R.: A Neural Network for Evaluating King Danger in Shogi, 第7回ゲーム・プログラミングワークショップ (GPW2002), Kanagawa, Japan, pp. 36–43 (2002).
7) Grimbergen, R. and Matsubara, H.: Pattern Recognition for Candidate Generation in the game of Shogi, *Proceedings of the Workshop on Computer Games (W31) at IJCAI-97*, Nagoya, Japan, pp. 7–12 (1997).
8) Nagai, A.: A new AND/OR Tree Search Algorithm Using Proof Number and Disproof Number, *Proceedings of Complex Games Lab Workshop*, Tsukuba, Japan, ETL, pp. 40–45 (1998).
9) Nagai, A. and Imai, H.: Proof for the Equivalence Between Some Best-First Algorithms and Depth-First Algorithms for AND/OR Trees, *IEICE Trans. Inform. & Systems*, Vol. E85-D, No. 10, pp. 1645–1653 (2002).
10) Platt, J.: Probabilistic outputs for support vector machines and comparison to regularized likelihood methods, *Advances in Large Margin Classiers* (A.J. Smola, P. Bartlett, B. S. D. S.(ed.)), MIT Press, pp. 61–74 (2000).
11) rnkranz, J. F.: Machine Learning in Games: A Survey, *Machines that Learn to Play Games* (Fürnkranz, J. and Kubat, M.(eds.)), Nova Science Publishers, Huntington, NY, chapter2, pp. 11–59 (2001).
12) Sakuta, M. and Iida, H.: The Performance of PN*, PDS and PN Search on 6x6 Othello and Tsume-shogi, Technical report, CGRI (1999).
13) Yoshimasa Tsuruoka, D. Y. and Chikayama, T.: Game-tree Search Algorithm based on Realization Probability, *ICGA Journal*, Vol. 25, No. 3, pp. 145–152 (2002).
14) 薄井克, 鈴木豪, 小谷善行: プロの棋譜を用いたTD法による将棋の評価関数の学習, 技術報告, ゲーム情報学 (2000).
15) 三輪誠, 横山大作, 田浦健次朗, 近山隆: SVMを用いた将棋の詰みの有無の予測の学習, 情報処理学会第66回全国大会 (2004).
16) 金子知適, 田中哲朗: 将棋プログラムにおける棋譜を利用した囲いの評価, 夏のプログラミングシンポジウム予稿集 (2003).
17) 金子知適, 田中哲朗, 山口和紀, 川合慧: 駒の関係を利用した将棋の評価関数, 第8回ゲーム・プログラミング ワークショップ, pp. 14–21 (2003).
18) 川田敦史: 将棋の局面情報の学習に関する研究, 修士論文, 名古屋工業大学 (2001).