合議のための多様な将棋プレイヤの集団学習

鈴木洋平[†1] 三輪誠[†2] 金田康正[†1,†3]

近年，複数の将棋プレイヤによる多数決を行うことによって最終的な指し手を決定する，合議と呼ばれる手法が注目されている．合議とは，構成する複数のプレイヤを評価関数の評価値を乱数を用い変化させることによって生成する手法である．本稿では，集団学習手法の一つである Bagging を用いてそれぞれのプレイヤの評価関数を異なる学習例を用いて学習させ複数のプレイヤを生成する手法を提案する．さらに，プレイヤの多様性の向上を目的とした学習について提案する．提案手法を既存のコンピュータ将棋プレイヤに適用した結果，既存の乱数を用いた合議に対し 56.3%という勝率で既存手法に対し有意に勝ち越すことに成功した．

Ensemble learning of diverse Shogi players for consultation algorithm

Yohei Suzuki,[†1] Makoto Miwa[†2] and Yasumasa Kanada[†1]

Consultation algorithm for shogi players has been recently focused. The algorithm creates various players from a single player by adding small random values to its evaluation function scores, and decides a move by applying majority rule to the players' decision. This paper proposes a method to apply an ensemble learning method Bagging to computer shogi players; it constructs different players by training their evaluation functions with different training examples. This paper then explore a way to increase the variety of shogi players. The shogi player by the proposed methods was statistically significantly better the player by the consultation algorithm in a 56% winning rate.

1. はじめに

近年，疎結合並列環境を用いた将棋プレイヤの判断決定の方法として，合議が注目されている[1]．合議には，比較的単純な実装方法で並列化を実現できる，多数決により判断ミスの少ない強い将棋プレイヤを生成できるという利点がある．小幡らは将棋プログラム bonanza に，乱数合議と呼ばれる手法を用いることによって従来の bonanza に有意に勝ち越すプレイヤを生成することに成功した[1)2)]．なお，乱数合議とは，将棋プレイヤが持つ評価関数の評価値を乱数により変化させることで複数の判断を生成し，それらを用いた多数決などで最終的な指し手を決定する手法である．

複数の異なるプレイヤを生成して利用する手法は機械学習の分野において集団学習として広く研究されてきた．集団学習は，複数のプレイヤをそれぞれ異なる学習例を用いて学習することにより生成し，その判断を統合する手法である．Bagging[3]，Boosting[9] などの具体的な手法が存在し，パターン認識分野において多くの成果を出している[3)4)5)9)]．この手法は複数の異なるプレイヤを生成しその判断を統合することで，単一プレイヤのみの場合より判断のミスが減ることが期待できる．さらに，プレイヤ生成の際の学習例のサンプリング方法，プレイヤの学習方法等様々な観点からプレイヤ強化の議論が行えるのも利点の一つである．

様々な集団学習手法が議論されているが，近年注目を集めている集団学習を効果的にする方法の一つが，プレイヤの多様性に注目した集団学習である．一般に集団学習は，集団を構成する一つ一つのプレイヤが多様であるほうが効果的であると言われている[4]．そこで，プレイヤを学習する段階において，それぞれのプレイヤが異なるものになるよう学習を調整する手法が提案され，クレジットカードの査定に関する回帰問題に適用した実験ではその効果が実証されている[5]．

将棋プログラムにおいて，ボナンザメソッド[7] による棋譜を用いた評価関数の学習が一般的に行われている．集団学習手法の将棋に対する適用は可能である．将棋に対して集団学習を適用すれば，評価関数の値を乱数ではなく，学習の枠組みの中で変化させることが可能になる．

†1 東京大学工学系研究科
Graduate School of Engineering, The University of Tokyo
†2 マンチェスター大学コンピュータ科学科
School of Computer Science, The University of Manchester
†3 東京大学情報基盤センター
Information Technology Center, The University of Tokyo

本稿では，合議の各プレイヤの多様性に注目しながら，将棋プログラムに集団学習を適用し，強力な将棋プレイヤを生成する方法について提案する. 具体的には，Bagging に加え合議を構成する各プレイヤの分散を考慮した特別な目的関数を用いて多様性が増すようにしながら複数のプレイヤを作成し，合議させる方法を提案する.

本論文では以降，2 章において関連研究について述べたのち，3 章で提案手法について述べ，4 章では提案手法を用いての評価について述べ，5 章でその結果を考察する. 最後に 6 章でまとめと今後の課題を述べる.

2. 関 連 研 究

2.1 乱 数 合 議

乱数合議とは，複数の将棋プレイヤの判断を統合する合議手法の一つであり，単一の思考プログラムを用いて複数のプレイヤを作りたいときに有効な合議の実現法である[1]. 複数のプレイヤを生成する方法は，将棋プログラムにおける評価関数の探索局面に対する評価値を，個々のプレイヤごとに異なる乱数系列を与えることによって変化させ，各プレイヤのゲーム木探索の結果を異なるものにするというものである. 評価関数に与える乱数は、正規分布 $N(0, D^2)$ にしたがって生成し，局面のハッシュキーに割り当てる. これにより，個々のプレイヤの評価関数は同じ局面に対しては常に同じ評価値を算出する. このようにして生成された複数のプレイヤによる多数決 (あるいは最も高い評価値を算出したプレイヤの指し手[2]) で指し手を決定する.

小幡らは Bonanza[7] に対しこの乱数合議法を適用し，一手あたりのノード数を固定した上で単一の通常の Bonanza と対戦させ，合議のプレイヤ数や正規乱数の分散値によっては有意に通常の Bonanza に対し勝ち越すことを確認した[1].

2.2 集 団 学 習

集団学習とは，機械学習手法の一つであり，複数の判別を行うユニット (以降は判別器と呼ぶ) を生成し，それらの判断を統合した上で最終的な判断を生成する手法のことである. 手順としては，まず学習の段階において，複数の判別器を異なる手段により学習することで生成する. 次に，実際の問題に対する判断を生成する際，複数の判別器による出力を適当なルールによって統合し，最終的な判断を行う.

複数の判別器を利用する利点はいくつか存在するがそのうちの三点を述べる. 一つは， 異なる複数の意見を統合することで単一の判別器では誤ってしまうような場面を避けられる可能性があることである. 例えば，5 つのそれぞれが異なる判別器を用意し，それらの出力を多数決することによって最終的な出力を決定するとする. すると，3 つ以上の判別器が誤判別を行わない限りは最終的な出力が誤りになることはないので出力が安定する. 二点目の利点は，複雑な標本空間を分割し，各プレイヤに割り当てることが可能な点である. 三点目の利点は，個々のプレイヤの欠点を補い合うことができる点である.

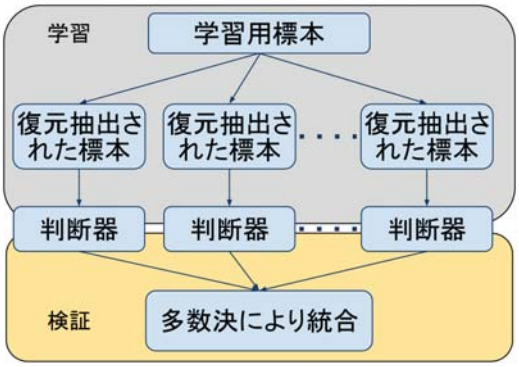

図 1 Bagging の仕組み

集団学習の手法を設計する際考慮すべき点は主に二つある. どのように異なる複数の判別器を生成するか，どのように生成された各判別器の出力を統合し最終的な出力にするかの二点である. これらの観点から考え出された，様々な集団学習手法が現在提案されているが，Bagging もそのような手法の一つである.

2.2.1 Bagging

Bagging とは，Bootstrap Aggregating の略であり，ブートストラップ法により生成された分類器を結合するというアルゴリズムである. その手順は図 **1** のようになる. ブートストラップ法とは，与えられた学習用標本集合から複数の標本集合の複製を生成する手法である. 具体的には，与えられた標本集合から，重複を許して標本のサンプリングを行い，新たな標本集合を構成する. つまり，元の標本集合の標本が何度も新しい標本集合の要素に選ばれてしまう可能性がある一方で，逆に一度も選ばれない標本があるかもしれない. この方法によって少し異なる複数の標本が得られる (図 1 の復元抽出された標本). それらを用いてそれぞれ判別器を学習を行い，その結果生成された複数の分類器の出力を多数決により統合する. 単純に感じられるが，この方法によって多くの場合，単一の判別器よりも高い精度が得られる. 理由としては，学習結果の分散が減ることが挙げられる. ここで分散とは，判別器の学習に用いる標本の違いが生成される判別器にどの程度影響してしまうかを表す. 一般に，Bagging は不安定な学習プログラムに有効であるとされているのもこの理由からである. また，学習用データの少ない場合も，ブートストラップ法によりデータサイズを維持したまま複数の学習用標本集合を生成できるので，Bagging が有用である.

2.2.2 Negative Correlation Learning

Negative Correlation Learning(NCL)[5] とは，最急降下法による学習において，特別な目的関数を用いた学習を行うことで，生成される複数の判別器の多様性に注目した学習を行うという，集団学習手法の一つである. Bagging では，異なる判別器が生成されるかどうかは復元抽出による学習標本の違いに依存したが，NCL では明示的に学習において判別器が多様になるよう調整する. NCL の手順は以下のようになる. はじめに、複数の判別器 (以後アンサンブルと呼ぶ) の出力 $\hat{f}$ を式 (1) のように表現する.

$$\hat{f}(k) = \frac{1}{\mathrm{M}} \sum_{i=1}^{\mathrm{M}} \hat{f}_i(k) \tag{1}$$

ここで、M はアンサンブルを構成する判別器の数，$\hat{f}_i(k)$ は i 番目の判別器の k 番目の訓練用標本に対する出力を表す. 次に最急降下法で個々の判別器を学習する際に用いる損失関数を式 (2)，式 (3) のように定義し，式 (2) を最少化することを考える.

$$e_i(k) = \frac{1}{2}(\hat{f}(k) - c(k))^2 + \lambda p_i(k) \tag{2}$$

$$p_i(k) = -(\hat{f}_i(k) - \hat{f}(k))^2 \tag{3}$$

ここで，$c(k)$ は k 番目の訓練標本に対する教師例，$p_i(k)$ はアンサンブルの分散が小さいことに関するペナルティ，λ はペナルティの影響度を意味する係数を表している. λ が大きくなるほど個々の判別器の精度は落ちるが，判別器の多様性は増すというトレードオフが存在する.

以上のように目的関数を定義し学習を行うことによって多様性に注目した集団学習を実現している．NCL をニューラルネットワークに適用した場合，扱う問題や条件によっては Bagging などの代表的な集団学習手法に勝る結果が得られている[6].

3. 提 案 手 法

3.1 将棋プログラムに対する Bagging の適用

集団学習はアルゴリズムの設計に際して，学習用標本数，プレイヤ数，個々のプレイヤの複雑さ，モデルの多様性，個々の学習用標本の重要度，判断を統合する方法など，様々な観点が存在し，それらのパラメタの調整の手法が議論されている. パラメタの多さはアンサンブルの設計が複雑で曖昧になる危険をはらんでいる. しかし，調整する部分が多いことから，設計の目的に沿ったアンサンブルの調整を行うことが比較的容易にできる. さらに，Boosting など，手法によっては統計学的な観点から学習の精度の向上について議論することができる. 将棋プログラムに関する疎結合な並列化手法である合議の既存手法は，アンサンブルの調整が困難であった. 例えば，乱数合議は，調整の際工夫できる部分はどのような乱数を生成するか，どのように判断を統合するのか程度であるが，乱数とアンサンブルの強さを関連付けて説明することは困難であると考えられる.

近年，将棋プログラムの評価関数を生成する際，ボナンザメソッド[7] と呼ばれる熟練者の棋譜を教師例とする学習を行うことが一般的であり，集団学習を将棋プログラムに適用することは可能である. さらに，適切な設計方法を用いて集団学習を調整することができれば，より強力な合議プレイヤが生成できる可能性が高い.

本提案手法では，学習用の棋譜を用いて集団学習の代表的手法である Bagging による学習を行い複数の評価関数を生成し，その結果得られたプレイヤを用いて合議をする手法を提案する. さらに集団学習による合議の調整の一例として，プレイヤの多様性に注目した学習について提案する. 具体的には，用意された棋譜から，個々のプレイヤに対して重複を許して決まった数の棋譜を抽出する. 抽出された棋譜を用いてプレイヤの評価関数をボナンザメソッドを用いて学習する. これを必要なプレイヤの数だけ繰り返すことによって，複数の異なるプレイヤを生成する. 実際に対戦を行う際は，生成されたプレイヤを用いて多数決を行い，最も支持された指し手を最終的な指し手とする.

本提案手法では集団学習における標本の単位を一棋譜としたが，標本の単位を棋譜の中の一つ一つの局面，一つの局面に対する候補手，といったように細かくすることも可能である. しかし，標本が細かくなりすぎること，学習の際の処理が複雑になることから，本提案手法では標本の最小単位を棋譜とした.

3.2 将棋プログラムに対する NCL の適用

将棋プログラムに Bagging を適用した上で，さらに生成される複数のプレイヤの多様性が増すように学習を調整する. 一般に将棋プログラムの学習に用いる目的関数は式 (4) のようになり，これを最小化するように学習することを考える.

$$J(P_0, P_1, ..., P_{N-1}, \boldsymbol{v}) = \sum_{n=0}^{N-1} l(P_n, \boldsymbol{v}) \tag{4}$$

ここで，$J(P_0, P_1, ..., P_{N-1}, \boldsymbol{v})$ は，評価関数を学習する際の目的関数で，$\boldsymbol{v}$ はその重みである．P_n は学習の際に探索する候補手を，N はその候補手の数を表す. $l(P_n, \boldsymbol{v})$ は，局面における候補手と，棋譜の手の評価値の違いの度合いを測る関数である[7]. この目的関数を評価関数の重みベクトル $\boldsymbol{v}$ に関して偏微分し，得られた傾斜を用いて評価関数の重みを更新する.

本稿では，この目的関数にプレイヤの多様性に関する項を加えることによって，合議プレイヤをさらに強力にする方法を模索する. 通常の NCL では，式 (2)，式 (3) のように，アンサンブルの各プレイヤの多様性が増すように，各判別器の訓練標本に対する出力の分散をペナルティ項として用いていた. しかし，将棋プログラムは，各プレイヤの正確な出力（つまり評価値や，指し手) を導出するためにゲーム木を用いた探索を行なわなければならず，これをアンサンブルを構成

している各プレイヤに対し行えば膨大な計算量が必要になる. 浅い探索や，局面の静的評価を用いて式 (3) のようなペナルティ項を導入することはできるが，今回は簡単のためこれまで学習した評価関数との距離をペナルティとした. 以上のように，本提案手法は正確には NCL[5] と異なる. 以降は，区別のため本稿の手法を Model-based NCL(MNCL) と記す.

学習の手順としては以下のようになる. まず，ブートストラップ法による復元抽出により一番目の評価関数を学習するための標本を用意し，式 (4) の目的関数に従い評価関数を学習する. 二番目以降の評価関数を生成する際にも同じように復元抽出により学習用標本を用意するが，学習の際には式 (5) で表されるような目的関数を用いて評価関数を学習する.

$$J_i(P_0, P_1, ..., P_{N-1}, \boldsymbol{v_i}) = \sum_{n=0}^{N-1} l(P_n, \boldsymbol{v_i}) - \frac{\lambda}{2(i-1)} \sum_{j=1}^{i-1} (\boldsymbol{v_i} - \boldsymbol{v_j})^T (\boldsymbol{v_i} - \boldsymbol{v_j}) \quad (5)$$

ここで，$J_i(P_0, P_1, ..., P_{N-1}, \boldsymbol{v_i})$ は，評価関数を学習する際の目的関数で，$\boldsymbol{v_i}$ はその重みである. 式 (5) の 2 番目の項は，重み同士の距離に関する項であり，λ それをどの程度重視するかを表す係数である. 式 (5) の目的関数を $\boldsymbol{v_i}$ について偏微分すると式 (6) のようになる.

$$\nabla_{\boldsymbol{v_i}} J_i(P_0, P_1, ..., P_{N-1}, \boldsymbol{v_i}) = \nabla_{\boldsymbol{v_i}} (\sum_{n=0}^{N-1} l(P_n, \boldsymbol{v_i})) - \lambda(\boldsymbol{v_i} - \overline{\boldsymbol{v}}) \quad (6)$$

$\overline{\boldsymbol{v}}$ は $i-1$ 番目までの評価関数の重みを平均したものである. この式を評価関数の更新の際に用い，二番目以降のプレイヤの評価関数を，以前に生成された評価関数からの距離が離れるように学習する. その結果得られた複数のプレイヤを用いて，実際の対戦において多数決合議を行い，指し手を決定する.

4. 評　　価

4.1 評 価 設 定

4.1.1 学習の設定

本手法の有用性を示すために，将棋プログラム「激指[8]」を用いて実験を行った. 激指は，第 20 回世界コンピュータ選手権において優勝を収めている，トップレベルの将棋プログラムの一つである. 本研究でプレイヤを学習する際は，激指において用いられている評価関数の特徴を用いる. 学習前の評価関数の重みの初期値は単純な駒割りの値（例えば，持ち歩の価値は 100 とする，など) のみとし，これに対し学習を行うことによって性能評価を行った.

本提案手法では，評価関数の学習にボナンザメソッド[7] を用いている. ボナンザメソッドは，熟練者の棋譜を元に評価関数のパラメタを調整する方法である. 棋譜中の各局面において，正解となる熟練者の手とそれ以外の候補手が存在する. それぞれの候補手についてゲーム木を探索した結果得られた評価値について，熟練者の手がその他の候補手よりも高く評価されるように，式 (4) の目的関数の，評価関数の重みに関する傾斜を用いてパラメタの調整を行う. 本提案手法では，ボナンザメソッドによる学習の際のゲーム木の探索深さを 6 として実験を行っている. また，学習用に熟練者の棋譜 20,000 棋譜を用意し，学習に用いる棋譜の数をのべ 40,000 棋譜とする. つまり，20,000 棋譜から 40,000 棋譜を復元抽出して学習実験を行う. このように学習用の棋譜数を固定する理由は，計算量をなるべく公平にするためである.

4.1.2 対戦実験の設定

対戦の初期局面は用意したテスト用棋譜の最初の 30 手が進行した後の局面とし，ひとつの初期局面に対し先手後手を入れ替えて対戦することを繰り返し勝率を求めた. 一手あたりの持ち時間は 3 秒に固定した.

4.1.3 比較用の単一のプレイヤの生成

集団学習と通常の学習の精度を比較するため，ボナンザメソッドを用いて 40,000 棋譜を学習し，単一のプレイヤを用意した. 以降，この単一のプレイヤを単に比較手法と呼ぶ.

4.2 乱 数 合 議

激指のプログラムに乱数合議を適用し，その強さを確認した. 乱数によって比較手法の評価関数を変化させプレイヤの数だけの計算機を用いた合議プレイヤと，一台の計算機上で単一の比較手法の評価関数を用いたプレイヤと対戦させた. 後に集団学習手法との比較するため，同じのべ 40,000 棋譜を学習した比較手法を乱数で変化させることにした. また，合議を構成するプレイヤ数は 10 とする. 集団学習を構成するプレイヤ数を 10 に固定したことにあわせて，乱数合議のプレイヤ数を定めた.

探索局面に対する評価値に加える正規乱数 $\mathrm{N}(0, \mathrm{D}^2)$ の D 値は，50，75，100，125，150，175，200 とした. D 値は，小幡らの設定に従い，激指の歩一枚の価値が 110 程度であるので，その周辺の値を利用した. 対戦実験では，500 局ずつ対戦した.

対戦の結果は**表 4.2** のようになる. 太字は，乱数合議が二項検定において 0.05 の危険度で有意に勝ち越していることを表している. 表を見ると，多くの条件で乱数合議が勝ち越している. このことから，激指においても乱数を用いた合議手法は有効であることがわかる. 特に，D の値を 175 としたときには 60% を越える勝率をあげることがわかった. 以降，単に乱数合議と記した場合は D の値が 175 で，プレイヤ数 10，比較手法の評価関数を用いた乱数合議手法のことを指す.

4.3 将棋への Bagging の適用

激指に Bagging を適用して学習を行い，得られた

表 1　乱数合議の単一プレイヤに対する勝率

D 値	50	75	100	125	150	175	200
比較手法に対する勝率	48.1%	**56.1%**	51.1%	**58.3%**	**56.3%**	**60.1%**	53.8%

初期パラメタ:
学習に用いる棋譜の絶対数 N
学習に用いる棋譜ののべ数 M
プレイヤ数 P
各プレイヤに共通な学習に用いる棋譜数 K(< M)

学習:
(1) 熟練者の棋譜 N 棋譜から M 棋譜を復元抽出する.
(2) すべてのプレイヤに共通な学習
  (a) M 棋譜のうち K 棋譜を使用
  (b) ボナンザメソッドを用いた学習によって，評価関数を一つ生成する.
(3) 各プレイヤによって異なる学習
  (a) (M-K) 棋譜を使用
  (b) 各プレイヤの評価関数を $\frac{M-K}{P}$ 棋譜を用いて学習する. このとき，初期値は (2) の際に生成された評価関数とする.

テスト:
- 学習の結果得られた P 個のプレイヤがそれぞれ探索によって指し手を決定. それらをもとに多数決によって最終的な指し手を決定.

図 2　将棋への Bagging の適用の手順

合議プレイヤの精度を確かめた. 手順は**図 2** のようになる. 本実験では比較手法と同じように N=20,000, M=40,000 として実験を行う. さらにプレイヤ数 P=10 とした. プレイヤ数も集団学習において重要な要素といえるが，本提案手法は学習用標本と学習の目的関数を調整することによって合議を強化することを目的としているので，今回は固定した. プレイヤ数を 10 に固定した理由は，図 2 におけるプレイヤあたりの学習用棋譜数の計算がやりやすいため，及びあまりプレイヤ数が少なすぎないためである.

通常の Bagging 手法では，図 2 の (2) の部分が存在しない. つまり，復元抽出により得られた標本を単にプレイヤ数で等分し，それぞれの学習にあてる. 将棋プログラムに Bagging を適用するにあたって，すべてのプレイヤに共通で学習する部分を導入した理由は，各プレイヤが学習に用いる棋譜数にある. 今回比較手法は 40,000 棋譜を学習に用いているが，集団学習手法では一つのプレイヤを学習するために必要な棋譜数が比較手法に比べて極端に少なくなってしまう. 例えば K=0 とした場合，一つのプレイヤの学習に用いる棋譜数は $40,000/10 = 4,000$ 棋譜とあまりに少ない棋譜数になってしまう. 一方，K=20,000 とした場合一つのプレイヤあたりの学習用棋譜数は $20,000 + (40,000 - 20,000)/10 = 22,000$ となり，評価関数を学習するのに十分な棋譜数が得られる.

以上のような手法を用いて学習を行い，生成された合議プレイヤと比較手法の対戦実験を行った. K の値を変化させてそれぞれ学習を行った. その結果得られた合議プレイヤと，比較手法を 200 局対局させた結果を**表 2** に示す. 太字は危険度 0.05 の二項検定で有意に勝ち越していることを意味する.

実験の結果 K=25,000 のとき比較手法に有意に勝ち越すことがわかった.K の値が小さいときには比較手法に負け越しているが，これはプレイヤあたりの学習棋譜数が少ないことが原因と考えられる. また，K=30,000，K=35,000 など，K の値が大きい場合にも勝率が下がる傾向がみられるが，これは各プレイヤが個別に学習する棋譜の数が少なくなり，結果生成される複数のプレイヤの多様性が落ち，すべてのプレイヤが同じような判断しか行わなくなったことが理由と考えられる.

最も比較手法に対する勝率が高かった K=25,000 の場合について，乱数合議と対戦実験を 500 局おこなった. すると，勝率は 51.3% となった. このことから，Bagging を将棋プログラムに適用して合議を行えば，乱数合議に有意に勝ち越すとはいえないものの，互角に戦うことができることがいえる.

4.4　将棋プログラムへの MNCL の適用

将棋プログラムの集団学習を，さらにプレイヤの多様性に注目するよう調整することで強化を試みる. 手順は**図 3** のようになる. 基本的には図 2 と同じだが，学習の (3) の段階で用いる目的関数を通常の式 (4) ではなく，式 (5) に従うものとしている. すなわち，各プレイヤの評価関数の重み同士の距離が離れるように学習するようにする.

パラメタ調整の際に重要になるパラメタは，式 (5) において，評価関数の分散に関するペナルティ項に乗じられている係数 λ である. 本実験では，表 2 の実験で設定した各 K 値について，$\lambda = 10^{-5}$，10^{-6} の場合で学習し，比較手法との対戦実験を行った. 本来 λ の値の決定は繊細で困難な作業であるが，本提案手法では大体の値を設定し，λ 値の導入により合議に何がもたらされるのかを注目することに重点を置く. $\lambda = 10^{-5}$，10^{-6} に設定した理由は，$\lambda = 10^{-4}$ とすると評価関数が極端にお互いの距離をとるように学習してしまい，評価関数のパラメタが発散してしまうので有効な値とはいえず，まだ $\lambda = 10^{-7}$ では更新への影響がほとんどないと考えたためである. 対戦実験は持ち時間 3 秒

表 2　Bagging により生成した合議プレイヤの単一プレイヤに対する勝率

K の値	0	5,000	10,000	15,000	20,000	25,000	30,000	35,000
プレイヤが個別に学習する棋譜数	4,000	3,500	3,000	2,500	2,000	1,500	1,000	500
各プレイヤの学習した棋譜数の合計	4,000	8,500	13,000	17,500	22,000	26,500	31,000	35,500
比較手法に対する勝率	45.6%	50.0%	47.8%	56.4%	51.8%	**58.1%**	53.1%	52.7%

表 3　Bagging により生成した合議プレイヤの単一プレイヤに対する勝率

K の値	0	5,000	10,000	15,000	20,000	25,000	30,000	35,000
プレイヤが個別に学習する棋譜数	4,000	3,500	3,000	2,500	2,000	1,500	1,000	500
各プレイヤの学習した棋譜数の合計	4,000	8,500	13,000	17,500	22,000	26,500	31,000	35,500
$\lambda = 10^{-5}$	40.8%	55.1%	44.9%	51.0%	52.1%	52.9%	56.1%	53.7%
$\lambda = 10^{-6}$	45.2%	42.3%	52.8%	**58.0%**	52.1%	**62.1%**	50.8%	42.9%

初期パラメタ:
学習に用いる棋譜の絶対数 N
学習に用いる棋譜ののべ数 M
プレイヤ数 P
各プレイヤに共通な学習に用いる棋譜数 K($< $ M)
プレイヤの分散の程度を決定する係数 λ

学習:
(1)　熟練者の棋譜 N 棋譜から M 棋譜を復元抽出する.
(2)　すべてのプレイヤに共通な学習
　(a)　M 棋譜のうち K 棋譜を使用
　(b)　ボナンザメソッドを用いた学習によって，評価関数を一つ生成する.
　(c)　**通常の目的関数を用いて学習する.**
(3)　各プレイヤによって異なる学習
　(a)　(M-K) 棋譜を使用
　(b)　各プレイヤの評価関数を $\frac{M-K}{P}$ 棋譜を用いて学習する. このとき，初期値は (2) の際に生成された評価関数とする.
　(c)　**プレイヤの多様性に関する項を加えた目的関数を用いて学習をする. 多様性の度合いを λ によって調整する.**

テスト:
- 学習の結果得られた P 個のプレイヤがそれぞれ探索によって指し手を決定. それらをもとに多数決によって最終的な指し手を決定.

図 3　将棋への MNCL の適用の手順

で，各 200 局行った. 結果を**表 3** に示す. 太字は有意に勝ち越していることを意味する.

K=15,000 より大きい場合は，$\lambda = 10^{-5}$，10^{-6} の条件のどちらかで常に同じ条件の Bagging 手法より比較手法に対し勝ち越している. 特に，K=25,000，$\lambda = 10^{-6}$ で学習を行った際には 62.1%の勝率で比較手法に勝ち越している. 逆に K の値が小さいときは大きく負け越すことが多い. これは，合議を構成する各プレイヤの評価関数の重みの距離が離れるように多くの棋譜を学習し続けた結果，パラメタが大きく発散してしまった結果であると考えられる.　$\lambda = 10^{-5}$ として学習を行った場合，大きく合議プレイヤが強力になることはないが，K=30,000,K=35,000 など，各プレイヤが共通に学習する棋譜数が多い場合は，同じ条件の Bagging 手法や $\lambda = 10^{-6}$ の場合より高い勝率になった. これは，プレイヤが個別に学習する棋譜の少なさを，λ を大きく設定し強く評価関数を分散させることで補ったと考えられる.

最も比較手法に対する勝率が高かった K=25,000，$\lambda = 10^{-6}$ の場合について，乱数合議と対戦実験をおこなった. 条件は持ち時間 3 秒で，500 局とした. すると，勝率は 56.3%となった. これは危険度 0.05 の二項検定で有意に勝ち越している. このことから，将棋プログラムにプレイヤの多様性に注目しつつ集団学習を適用すると，パラメータ調整によっては既存の合議手法にも有意に勝ち越すようなプレイヤを生成することができるといえる. 比較手法，乱数合議，Bagging による集団学習，MNCL による集団学習が持ち時間 3 秒で 500 局対局したときの勝率を**表 4** にまとめる. この表からも MNCL を用いた集団学習手法が強力であることがわかる.

5. 考　　察

5.1　合議の多様性と学習の精度に関する考察

実際に特殊な目的関数を用いて学習することでプレイヤの多様性は増しているのだろうか. また，自己対戦以外の条件でも正しい判断が行えるだろうか. テスト用に 500 棋譜と用意し，各集団学習手法に対し以下の値を算出した.

(1)　棋譜に対する一致率 (%)
(2)　一局面ごとに挙がる候補手の平均 (候補手平均)
(3)　候補手数ごとの意見の分かれ具合 (候補手散乱)

一致率は局面に対する棋譜の手と，合議プレイヤの決

表 4　各手法の比較 (ヘッダ行の手法に対する勝率)

	比較手法	乱数合議	Bagging	MNCL
比較手法		39.9%	41.7%	37.6%
乱数合議	**60.1%**		48.7%	43.7%
Bagging	**58.3%**	51.3%		44.5%
MNCL	**62.4%**	**56.3%**	**55.5%**	

表 5　各集団学習手法の一致率と多様性に関する指標の値

K の値		5,000	10,000	15,000	20,000	25,000	30,000	35,000
単純な Bagging 手法	一致率	38.3%	38.4%	38.9%	38.4%	38.7%	38.7%	38.8%
	候補手平均	1.58	1.57	1.63	1.60	1.66	1.66	1.73
	候補手散乱 (C=2)	21.0	20.4	23.6	21.7	25.4	25.4	28.4
	候補手散乱 (C=3)	17.8	17.3	19.5	18.1	20.1	20.1	21.9
MNCL($\lambda = 10^{-5}$)	一致率	38.8%	38.9%	38.9%	38.8%	38.9%	38.8%	38.9%
	候補手平均	1.58	1.60	1.62	1.65	1.66	1.68	1.72
	候補手散乱 (C=2)	20.5	22.0	23.5	24.4	25.3	26.5	27.9
	候補手散乱 (C=3)	17.6	18.7	19.4	19.8	20.1	20.8	21.5
MNCL($\lambda = 10^{-6}$)	一致率	38.8%	39.0%	39.0%	38.9%	38.9%	38.8%	38.9%
	候補手平均	1.58	1.60	1.62	1.64	1.66	1.69	1.72
	候補手散乱 (C=2)	20.5	22.0	23.5	24.2	25.0	26.2	27.9
	候補手散乱 (C=3)	17.6	18.7	19.0	19.6	20.4	21.2	21.4

断がどの程度一致しているかを百分率で表現している. 候補手平均は，合議の各プレイヤが探索により導き出した手の種類を局面ごとに平均したものである. 候補手散乱は，候補手数ごとの意見の分かれ具合を表す指標で，式 (7) のように算出した.

$$\mathrm{GogiScat(C)} = \frac{\sum_{i=1}^{\mathrm{N}} \sum_{j=1}^{\mathrm{C}} (\frac{\mathrm{P}}{\mathrm{C}} - p_{ij})^2}{\mathrm{N}} \qquad (7)$$

ここで，C は候補手の数，N は学習する局面数，P は合議を構成するプレイヤ数，p_{ij} は i 番目の局面における j 番目の候補手を支持したプレイヤの数である. 意見がまったく等しく分かれた状態からの距離を表し，値が大きければ一方の候補手がより多く支持されていることを，値が小さければ僅差で指し手が決定したことを表す. 本研究では C = 2，3 の場合について式 (7) の値を算出した. 一定数以上の棋譜を用いて検証した場合，合議の各プレイヤが導出する候補手が 1 通りから 3 通りである場合が全体の 95%以上を占めることが多いことが C = 4 以降を算出しなかった理由である.

表 2，表 3 で用いた各条件について，以上の値を算出した結果を**表 5** に示す. なお，比較手法の一致率は 38.8%であった. 全手法に共通して，K の値が大きくなるにつれ，候補手平均と候補手散乱は大きくなった. 候補手散乱が K の値とともにに大きな値をとるのは，プレイヤが個別に学習する棋譜の数の少なさが，多様な評価関数の生成を妨げていることが原因と考えられる. 候補手平均が K の値とともに大きくなる理由は不明だが，候補手散乱も増大していることから，K の値が大きい場合，ごく一部が他のプレイヤと異なる手を支持するという事象が多く起きている可能性がある. つぎに手法ごとに表 5 の各数値を見ると，わずかではあるが MNCL による集団学習手法のほうが，Bagging に比べ一致率が高い. この一致率の高さが直接合議プレイヤの強さに影響しないことは表 2，表 3 により明らかであるが，熟練した人間の棋譜をうまく学習するという意味では Bagging よりも MNCL のほうが優れているといえる.

以上の新たな知見が得られたものの，MNCL を導入することでプレイヤが多様になることを強く支持するデータは得られなかった. 実際には提案手法がプレイヤを多様にするようには働いていない可能性，今回求めた数値では正確なプレイヤの多様性は測れない可能性が考えられる.

5.2　学習の収束性に関する考察

提案手法による合議プレイヤの学習により，強力な合議プレイヤを生成することに成功した. しかし，パラメータの設定によっては，比較手法に大きく劣る場合もある. このような場合の原因の一つとして，学習に用いる目的関数に各プレイヤの評価関数の距離を広げるような項を導入して学習を続けることによって，パラメタが発散してしまっている可能性が考えられる.

パラメタの発散を防ぐ目的で，評価関数の重みが大きくなりすぎないよう目的関数に新たな罰則項を加えることを考える. すると，目的関数は式 (8) のようになる.

$$\begin{aligned} J_i(P_0, P_1, ..., P_{N-1}, \boldsymbol{v_i}) = & \sum_{n=0}^{N-1} l(P_n, \boldsymbol{v_i}) \\ & - \frac{\lambda}{2(i-1)} \sum_{j=1}^{i-1} (\boldsymbol{v_i} - \boldsymbol{v_j})^T (\boldsymbol{v_i} - \boldsymbol{v_j}) \\ & + \frac{\mathrm{C}}{2} \boldsymbol{v_i}^T \boldsymbol{v_i} \end{aligned} \qquad (8)$$

C はベクトルの大きさに関するペナルティの大きさを決める係数である. この式を v_i について偏微分すると式 (9) が得られる.

$$\begin{aligned} \nabla_{\boldsymbol{v_i}} J_i(P_0, P_1, ..., P_{N-1}, \boldsymbol{v_i}) = & \nabla_{\boldsymbol{v_i}} (\sum_{n=0}^{N-1} l(P_n, \boldsymbol{v_i})) \\ & - \lambda(\boldsymbol{v_i} - \overline{\boldsymbol{v}}) \\ & + \mathrm{C}\boldsymbol{v_i} \end{aligned} \qquad (9)$$

式 (9) を用いて評価関数の更新を行い，実験をおこなった. 手順は図 3 と同様で，学習の (3) の段階のみ式 (9) に従う更新を行っている. この式を用いて $\lambda = 10^{-5}$ で K =15,000，20,000，25,000 の場合について学習を行い，比較手法との対戦実験を行った.C の値は λ と同じ 10^{-5} とした. 表 3 の実験において，$\lambda = 10^{-5}$ の条件で学習した合議プレイヤは大きく結果が良くなってはい

ない. しかし，表 2 の実験や表 3 の実験の $\lambda = 10^{-6}$ の場合の実験結果をみると，K=15,000，20,000，25,000 のときは目的関数の工夫によってはよい合議プレイヤを生成できる可能性がある. $\lambda = 10^{-5}$ として行った学習が大きく勝たない理由が評価関数の分散のしすぎにあるのならば，それを抑える項を導入することでその精度を向上できる可能性がある. 比較手法と持ち時間 3 秒で 200 局の対戦実験を行った. 結果を**表 6** に示す.

図 3 と比較すると，勝率が向上していることがわかる. この結果はパラメタの調整によってはさらに合議プレイヤが強力になる可能性があることを示唆している.

6. おわりに

本稿では，プレイヤの多様性に着目し特別な目的関数を用いつつ，将棋プレイヤの学習に集団学習手法を適用し，合議プレイヤを生成する手法の提案をおこなった. 実験では，はじめに Bagging を将棋プログラムの学習に対し適用して合議プレイヤを生成し，既存のボナンザメソッドによる学習によって生成された単一プレイヤや乱数合議との比較を対戦実験によりおこなった. さらに，Negative Correlation Learning の考え方を Bagging による将棋の集団学習に導入することによって生成された合議プレイヤを，対戦実験によって既存手法と比較した. 実験の結果，表 4 に示した通り，よく調整された Bagging による合議は乱数合議と同等の強さを持つこと，Bagging による集団学習にさらに Negative Correlation Learning の考え方を用いて調整した合議は，単一プレイヤと乱数合議の双方に有意に勝ち越すことを確認した. 本稿の結果は，集団学習により強い合議プレイヤを生成できることを意味し，今後様々な集団学習を用いた評価関数の学習が期待される.

今後の課題は二点挙げられる. 一つは，多様性に注目した学習の精度の向上である. 本研究では集団学習に用いるパラメタの多くを強い合理的な理由なく定めたが，例えば λ の値はもっと適当なものがある可能性がある. さらに，本研究では評価関数の重みベクトルの分散の項を学習に用いる目的関数に導入したが，局面評価値の違いなど，他にもプレイヤの多様性を表現するような項は考えられる. また，プレイヤの多様性をうまく表現する指標を設定する必要がある.

二つ目の課題は，提案手法のような Bagging や，Bagging に NCL を適用する手法の他にも，様々な集団学習手法が存在する. 工夫によって将棋プログラムに適用できる手法も存在するので，将棋に有効と見積もることができれば積極的に適用していくべきである. 本提案手法では，定めるべきパラメタが多いことから調整の困難さという問題があったが，例えば Boosting の代表的手法である AdaBoost[9] を適用すれば，ほとんどのパラメータはアルゴリズムによって決定されている. また，本研究では合議の意思決定手段を多数決としたが，他にも Borda Count や Behavior Knowledge Space など様々な手法が存在する．Bagging は集団学習手法のなかで最も単純な手法のひとつといえるので，さらなる工夫の余地があるといえる.

表 6 収束性を考慮した合議プレイヤの単一プレイヤに対する勝率

K の値	15,000	20,000	25,000
$\lambda = 10^{-5}$	55.0%	54.8%	55.1%

謝　辞

本研究の一部は文部科学省科学研究費助成金特定領域研究「情報爆発に対応する高度にスケーラブルなソフトウェア構成基盤」の環境である InTrigger を利用して行われた. ここに謝意を示す.

参考文献

1) 小幡拓弥， 杉山卓弥， 保木邦仁， 伊藤毅志. 将棋における合議アルゴリズム:既存プログラムを組み合わせて強いプレイヤを作れるか?. The 14th Game Programming Workshop， pp. 51-58， 2009.
2) 杉山卓弥， 小幡拓弥， 保木邦仁， 伊藤毅志. 将棋における合議アルゴリズム - 評価値を用いる効果について -. The 14th Game Programming Workshop， pp. 59-65， 2009.
3) Breiman， L. Bagging Predictors. Machine Learning 24， pp. 123-140， 1996.
4) Geoffrey R. Factors Affecting Boosting Ensemble Performance on DNA Microarray data. Neural Networks (IJCNN)， The 2010 International Joint Conference on， On page(s): 1 - 7, Volume: Issue: ， 18-23， 2010.
5) Y.Liu， X.Yao. Ensemble learning via negative correlation. Neural Networks，vol. 12，no. 10， pp. 1399-1404， 1999.
6) Lofstrom. T， Johansson. U， Bostrom. H. Comparing methods for generating diverse ensembles of artificial neural networks. Neural Networks (IJCNN)， The 2010 International Joint Conference on， On page(s): 1 - 6， Volume: Issue: ， 18-23 ，2010
7) 保木邦仁. 局面の学習を目指した探索結果の最適制御. The 11th Game Programming Workshop，pp. 78-83， 2006.
8) 将棋プログラム「激指」のページ. http://www.logos.ic.i.u-tokyo.ac.jp/gekisashi/.
9) Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on- line learning and an application to boosting. In Computational Learning Theory: Eurocolt '95, pages 23—37. Springer-Verlag, 1995.