コンピュータ将棋の概念を打ち砕き
進化させるべく三駒関係を封印する

高速な差分計算を特徴とし
三駒関係と同等の NPS を実現する
ディープラーニング評価関数を搭載

the end of genesis
T.N.K.evolution turbo type D

ザイオソフト
コンピュータ将棋サークル

野田久順　岡部淳　鈴木崇啓
那須悠　河野明男

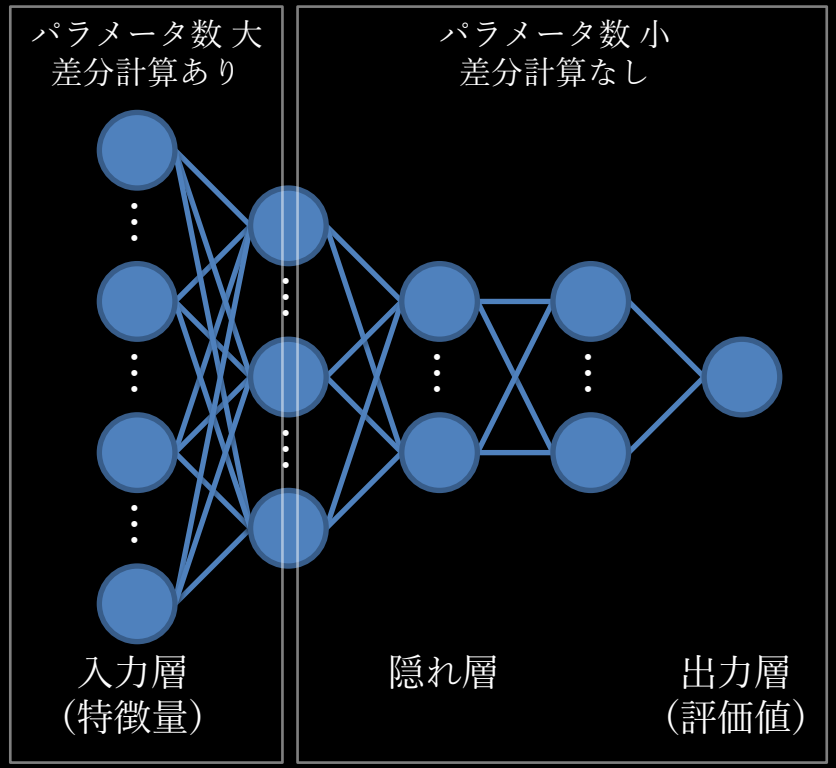
ベンチマーク

- 測定環境
  - やねうら王ベンチマーク
  - CPU: Core i7 6700K
  - 置換表サイズ: 16GB
  - スレッド数: 8
  - トーナメント版
  - NPSはやねうら王ベンチマークに収録されている3局面のNPSの平均値

CPUによる演算

- T.N.K. のディープラーニング評価関数はGPUを使わず、CPUで１局面ずつ評価します
  - αβ探索ベースの探索ルーチンにそのまま組み込むことができます
- パラメータを整数化し、SIMD演算で高速化しています

差分計算



- KPに相当する特徴量を入力とする全結合ニューラルネットワークです
- 入力のアフィン変換を差分計算で高速化しています
  - 二駒関係の差分計算をベクトルに拡張して適用しています

使用ライブラリ

- やねうら王
  - 用途
    - エンジンの
      基礎部分として使用
  - 選定理由
    - レーティングの高さ
    - 改造のしやすさ
- Apery
  - 用途
    - 学習データ生成時の
      評価関数として使用
  - 選定理由
    - レーティングの高さ

乞うご期待…

高速に差分計算可能なニューラルネットワーク型将棋評価関数

那須 悠 †

† ザイオソフト コンピュータ将棋サークル

2018 年 4 月 28 日

概要

現在のコンピュータ将棋ソフトの多くで使用されている評価関数である三駒関係は，高速な計算が可能であるが，非線形な評価を行うことができない．本稿では，CPU で高速に動作するニューラルネットワーク評価関数，「NNUE 評価関数」(Efficiently Updatable Neural-Network-based evaluation functions) を提案する．NNUE 評価関数は，CPU での演算に最適化した設計と，差分計算を始めとする高速化技術により，三駒関係と同等の実行速度で動作する非線形評価関数である．NNUE 評価関数を使用する初めての将棋ソフト『the end of genesis T.N.K.evolution turbo type D』は，第 28 回世界コンピュータ将棋選手権に参加を予定している．

NNUE: Efficiently Updatable Neural-Network-based Evaluation Functions for Computer Shogi

Yu Nasu†

†*Ziosoft Computer Shogi Club*

April 28, 2018

Abstract

Most of the strongest shogi programs nowadays employ a linear evaluation function, which is computationally efficient but lacks nonlinear modeling capability. This report presents a new class of neural-network-based nonlinear evaluation functions for computer shogi, called NNUE (Efficiently Updatable Neural-Network-based evaluation functions). NNUE evaluation functions are designed to run efficiently on CPU using various acceleration techniques, including incremental computation. The first shogi program with a NNUE evaluation function, *the end of genesis T.N.K.evolution turbo type D*, will be unveiled at the 28th World Computer Shogi Championship.

0. まえがき

本稿は，第 28 回世界コンピュータ将棋選手権 [1] に参加を予定しているソフト『the end of genesis T.N.K.evolution turbo type D』のアピール文書の一部である．同ソフトの評価関数に関する技術を論文風の体裁で説明するが，実験は掲載しない．同ソフトの具体的な棋力にも本稿では言及しないので，選手権の結果を参照されたい．

1. はじめに

コンピュータ将棋ソフトの指し手の決定は，主として形勢判断に相当する「評価関数」と，読みに相当する「探索」によって行われる．高い棋力を実現するためには，評価値の精度が高く，高速に計算できる評価関数を用いることが望ましい．評価値の精度が高ければ正確な形勢判断ができ，計算が高速であればより深く読むことができるためである．

2018 年 3 月現在，コンピュータ将棋ソフトで使用される最も代表的な評価関数の方式は，「三駒関係」と呼

ばれる線形モデルである．機械学習によって駒 3 つの位置関係に重みを付けておき，局面上に出現する位置関係に付けられた重みの総和を評価値とする．2003 年に初めてアイデアが示され [2]，2009 年に玉を含む組み合わせに限定して実装した『Bonanza』 [3] のソースコードが公開されて以降，広く用いられるようになった．2014 年に『NineDayFever』 [4] で導入された手番評価など，いくつかの拡張を経て，トップクラスの棋力を持つ将棋ソフトのほとんどで現在も採用されている．

三駒関係の優位性は，膨大な数のパラメータによって表現される評価関数でありながら，高速に評価値を算出できる点にある．ある局面から 1 手指した先の局面の評価値を求めるとき，移動した駒に関する重みを元の局面の評価値に足し引きする差分計算によって，全体を計算した場合と同じ結果を，遥かに効率的に求めることができる．差分計算の容易さは，評価関数に線形モデルを採用することによってもたらされる大きな利点である．

しかし，線形モデルであることは，評価関数の表現能力の足枷にもなっている．特定の駒の位置関係が，ある状況においては有利に働くが，別の状況では無駄になる，といった非線形な評価を，三駒関係では直接表現することができない．線形モデルのまま表現能力を高めるためには，評価項目を変える必要があるが， 2018 年 3 月現在，三駒関係より優れた線形モデルは知られていない．

より高い表現能力を持つ評価関数の実現を狙ったアプローチとして，非線形モデル，特にニューラルネットワークの利用も試みられている．先駆けとなったのは『習甦』 [5] で，自玉および相手玉の安全度によって駒の価値を重み付けした，非線形な評価関数を特徴とする将棋ソフトである．近年では，コンピュータ囲碁において，大規模な CNN (Convolutional Neural Networks) による評価関数を用いたソフトが成功を収めた [6] ことから，コンピュータ将棋でも CNN を利用する試みが行われている．しかし， CNN による評価関数を活用するためには，線形モデルよりも大きな計算リソースが必要となることが課題である．CNN を利用する将棋ソフトのほとんどは，並列演算を得意とする GPU を使用し，複数の局面をまとめて評価するバッチ処理を行って高速化している．それにもかかわらず， 2018 年 3 月現在，ニューラルネットワークに最適化された特殊なハードウェアを使用する『AlphaZero』 [7] を除いて，トップクラスの将棋ソフトに比肩する棋力を実現した例は報告されていない．

本稿では， CPU で高速に動作するニューラルネットワーク評価関数を提案する．CPU での演算に最適化した設計と，差分計算を始めとする高速化技術により，複数の隠れ層を持つニューラルネットワーク評価関数を，三駒関係と同等の実行速度で動作させることができる．以降，この方式の評価関数を「NNUE 評価関数」 (Efficiently Updatable Neural-Network-based evaluation functions) と呼ぶことにする．

2. NNUE 評価関数

NNUE 評価関数は， GPU もバッチ処理も用いず，CPU で 1 局面ずつ評価するニューラルネットワーク評価関数である．三駒関係と同様， 1 局面を評価するまでのレイテンシが小さく，枝刈りを伴うミニマックス系の探索手法と組み合わせやすい．

『the end of genesis T.N.K.evolution turbo type D』に搭載する NNUE 評価関数の模式図を図 1 に示す．CPU で高速に計算するために設計した構造であり，『AlphaZero』などで採用されているニューラルネットワークの構造とは大きく異なる．

以下では， NNUE 評価関数の設計と高速化技術について述べる．

2.1 全結合ニューラルネットワーク

将棋盤の二次元構造を利用する CNN ではなく，全結合ニューラルネットワークを使用する．主な利点は，メモリアクセスパターンが単純で高速化しやすいことと，後述する差分計算が容易に実現できることの 2 点である．

メモリアクセスパターンは，三駒関係より処理効率が高い評価関数を設計するために重要な要素である．三駒関係では，評価値を算出するためにかかる時間の大半を，メモリへのランダムアクセスが占めている．効率的なメモリアクセスパターンを用いれば，同じ時間でより複雑な計算が可能になる．

全結合ニューラルネットワークによる評価値の計算を次式に示す．入力となる特徴ベクトルを $\boldsymbol{x}$，隠れ層の数を L，層 l の重み行列を $\boldsymbol{W}_l$，層 l のバイアスベクトルを $\boldsymbol{b}_l$，活性化関数を σ とすると，評価値 y は

$$[y]_{1\times 1} = \boldsymbol{z}_{L+1} \tag{1}$$

$$\boldsymbol{z}_l = \boldsymbol{b}_l + \boldsymbol{W}_l \boldsymbol{a}_{l-1} \tag{2}$$

$$\boldsymbol{a}_l = \begin{cases} \sigma\left(\boldsymbol{z}_l\right) & (\text{if } l > 0) \\ \boldsymbol{x} & (\text{if } l = 0) \end{cases} \tag{3}$$

と求められる．

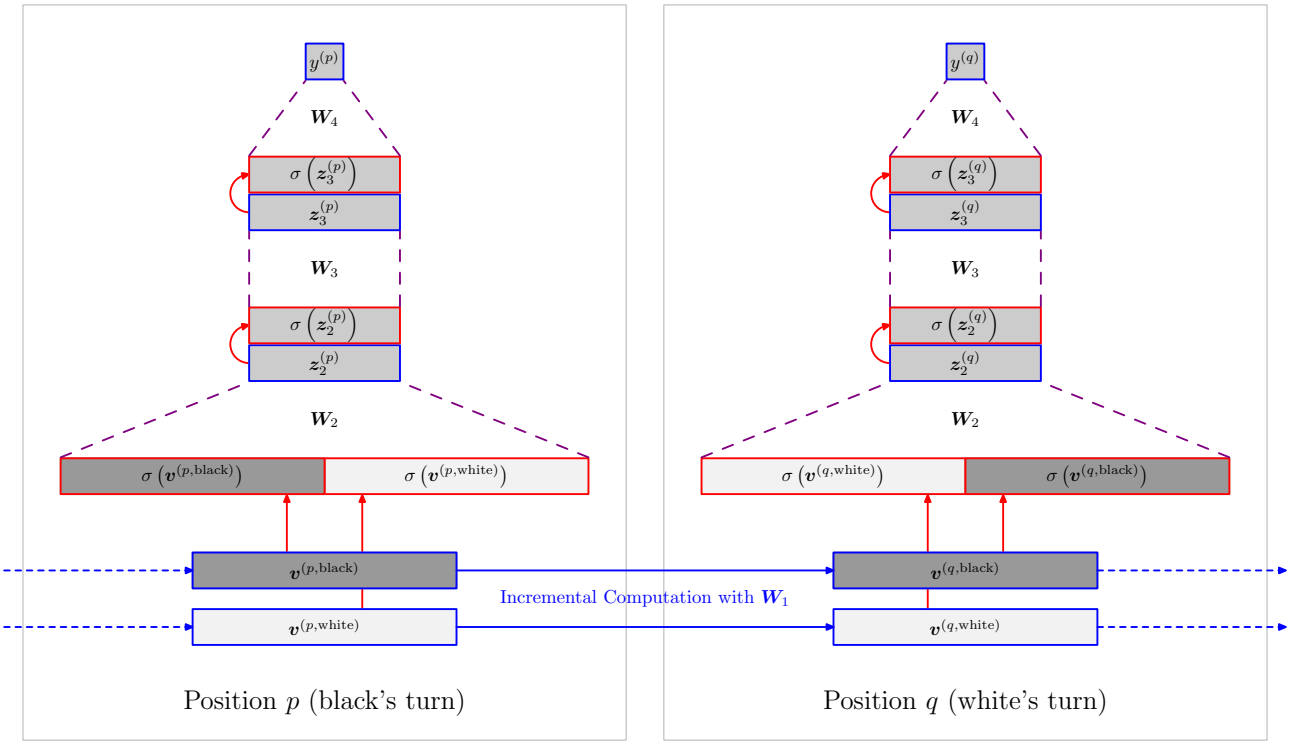


図 1　『the end of genesis T.N.K.evolution turbo type D』の評価関数．局面 p から 1 手指した先の局面 q について評価値を求めるとき，処理の一部で差分計算を利用する．評価には手番が考慮されている．

2.2　活性化関数

活性化関数には，層 l のユニット数を N_l として，次式で表される clipped ReLU を用いる．

$$\begin{aligned}\sigma\left(\boldsymbol{z}_l\right) &= \sigma\left(\begin{bmatrix} z_{l,1} \\ z_{l,2} \\ \vdots \\ z_{l,N_l} \end{bmatrix}_{N_l \times 1}\right) \\ &= \begin{bmatrix} \sigma(z_{l,1}) \\ \sigma(z_{l,2}) \\ \vdots \\ \sigma(z_{l,N_l}) \end{bmatrix}_{N_l \times 1}\end{aligned} \tag{4}$$

$$\sigma\left(z_{l,i}\right) = \begin{cases} 0 & (\text{if } z_{l,i} \leq 0) \\ z_{l,i} & (\text{if } 0 < z_{l,i} < 1) \\ 1 & (\text{if } z_{l,i} \geq 1) \end{cases} \tag{5}$$

値域が有限であるため整数化に適しており，後述する SIMD 演算によって効率的に計算できることが利点である．

2.3　特徴量

特徴ベクトルには，原則として，各要素が 0 または 1 の値を取る二値ベクトルを使用する．この制約も差分計算を容易にするためである．高速に差分計算を行うためには，1 手指す前後の局面で値が変化する要素の数が少ないことが望ましい．

このような性質を満たす特徴量の例としては，自玉または敵玉と，玉以外の駒 1 つとの位置関係を表す KP (King-Piece) が挙げられる．『the end of genesis T.N.K.evolution turbo type D』で実際に採用している特徴量は，KP をベースとして改変したものである．詳細は後述する．

2.4　アフィン変換とメモリレイアウト

式 (2) のようなアフィン変換を行うためには，通常，行列の行とベクトルの内積を計算するのが効率的である．$\boldsymbol{W}_l$ の i 行目を $\boldsymbol{W}_l(i,:)$ として，

$$\boldsymbol{z}_l = \begin{bmatrix} z_{l,1} \\ z_{l,2} \\ \vdots \\ z_{l,N_l} \end{bmatrix}_{N_l \times 1} \tag{6}$$

$$z_{l,i} = b_{l,i} + \boldsymbol{W}_l(i,:)\boldsymbol{a}_{l-1} \tag{7}$$

と計算する．行列の行の要素を連続して参照するメモリアクセスパターンであるから，メモリレイアウトは row-major とする．

しかし，ベクトル $\boldsymbol{a}_{l-1}$ がスパースである（要素の大

部分が 0 である）場合は，行列の一部の列だけを用いて，次式のように計算する方が高速になる．$\boldsymbol{W}_l$ の j 列目を $\boldsymbol{W}_l(:,j)$ として，

$$\boldsymbol{z}_l = \boldsymbol{b}_l + \sum_{j \in \{j | a_{l-1,j} \neq 0\}} a_{l-1,j} \boldsymbol{W}_l(:,j) \tag{8}$$

と計算する．この場合は，行列の列の要素を連続して参照するメモリアクセスパターンになり，メモリレイアウトを column-major とする方が効率が良い．

NNUE 評価関数では，特徴ベクトル $\boldsymbol{x}$ のアフィン変換

$$\boldsymbol{z}_1 = \boldsymbol{v} \tag{9}$$

$$\boldsymbol{v} = \boldsymbol{b}_1 + \boldsymbol{W}_1 \boldsymbol{x} \tag{10}$$

を，式 (8) の方法によって計算する．前述の通り，$\boldsymbol{x}$ は二値ベクトルである．$\boldsymbol{x}$ がスパースであれば，式 (8) によってアフィン変換を高速に計算できる[*1]．さらに，二値ベクトルであることを利用すると，式 (8) において $a_{l-1,j} = x_j \in \{0,1\}$ であるので，

$$\boldsymbol{v} = \boldsymbol{b}_1 + \sum_{j \in \{j | x_j = 1\}} \boldsymbol{W}_1(:,j) \tag{11}$$

と単純化できる．

2.5 差分計算

差分計算は，ある局面から 1 手指した先の局面の評価値を計算する際に，元の局面との差異が少ないことを利用して高速化する方法である．NNUE 評価関数では，局面 q の特徴ベクトル $\boldsymbol{x}^{(q)}$ をアフィン変換した結果のベクトル

$$\boldsymbol{v}^{(q)} = \boldsymbol{b}_1 + \boldsymbol{W}_1 \boldsymbol{x}^{(q)} \tag{12}$$

を，1 手前の局面 p について計算済みの $\boldsymbol{v}^{(p)}$ を利用して差分計算する．具体的には，次式によって $\boldsymbol{v}^{(q)}$ を求める．

$$\begin{aligned} \boldsymbol{v}^{(q)} = \boldsymbol{v}^{(p)} &- \sum_{j \in \left\{ k \middle| x_k^{(p)}=1 \wedge x_k^{(q)}=0 \right\}} \boldsymbol{W}_1(:,j) \\ &+ \sum_{j \in \left\{ k \middle| x_k^{(p)}=0 \wedge x_k^{(q)}=1 \right\}} \boldsymbol{W}_1(:,j) \end{aligned} \tag{13}$$

これは，線形モデルの差分計算をベクトルに拡張した式であると解釈することができる．

[*1] 実際には差分計算を行うので，1 手指す前後の特徴ベクトルの変化（ハミング距離）が小さいことが重要であり，スパースである必要はない．

差分計算を利用するのは特徴ベクトルのアフィン変換だけで，それ以外の部分では利用しない．

2.6 手番評価

将棋においては，駒の配置が同じであっても，手番がどちらにあるかによって形勢は異なるので，評価関数も手番を考慮する方が高精度になる．三駒関係では，線形性を利用して評価値を 2 つの項に分解し，手番に依存しない項と手番に依存する項の和で表現している [4]．

NNUE 評価関数では，常に手番側の視点から見た局面を評価することによって手番評価を実現する．まず，単純な方法について説明する．局面 p の手番側プレイヤーを active(p)，プレイヤー c から見た局面 p の特徴ベクトルを $\boldsymbol{x}^{(p,c)}$ として，

$$\boldsymbol{z}_1^{(p)} = \boldsymbol{b}_1 + \boldsymbol{W}_1 \boldsymbol{x}^{(p,\mathrm{active}(p))} \tag{14}$$

とすると，手番側の視点から見た局面の評価が実現できる．

手番がどちらにあるかによって特徴ベクトルが変わるため，差分計算は，先手（black）および後手（white）の両方の視点について行う必要がある．1 手指すごとに，$c \in \{\mathrm{black}, \mathrm{white}\}$ に対して，それぞれ

$$\boldsymbol{v}^{(p,c)} = \boldsymbol{b}_1 + \boldsymbol{W}_1 \boldsymbol{x}^{(p,c)} \tag{15}$$

を差分計算しておく．評価値の計算には手番側を使用し，

$$\boldsymbol{z}_1^{(p)} = \boldsymbol{v}^{(p,\mathrm{active}(p))} \tag{16}$$

とする．

次に，この方法を拡張して，非手番側のベクトルも評価値の計算に活用する方法を説明する．局面 p の非手番側のプレイヤーを opponent(p) として，式 (16) の代わりに次式を用いる．

$$\begin{aligned} \boldsymbol{z}_1^{(p)} &= \begin{bmatrix} \boldsymbol{v}^{(p,\mathrm{active}(p))} \\ \boldsymbol{v}^{(p,\mathrm{opponent}(p))} \end{bmatrix}_{2N_1 \times 1} \\ &= \begin{cases} \begin{bmatrix} \boldsymbol{v}^{(p,\mathrm{black})} \\ \boldsymbol{v}^{(p,\mathrm{white})} \end{bmatrix}_{2N_1 \times 1} & (\text{if active}(p) = \text{black}) \\ \begin{bmatrix} \boldsymbol{v}^{(p,\mathrm{white})} \\ \boldsymbol{v}^{(p,\mathrm{black})} \end{bmatrix}_{2N_1 \times 1} & (\text{if active}(p) = \text{white}) \end{cases} \end{aligned} \tag{17}$$

すなわち，手番側と非手番側の特徴ベクトルをそれぞれアフィン変換し，結合したベクトルを $\boldsymbol{z}_1^{(p)}$ とする．アフィン変換のパラメータ $\boldsymbol{b}_1, \boldsymbol{W}_1$ は共有されるので，こ

の計算は手番方向の畳み込みであると解釈できる．

図 1 に示した『the end of genesis T.N.K.evolution turbo type D』の評価関数は，式 (17) の方式を採用している．

2.7 HalfKP 特徴量

前述の通り，『the end of genesis T.N.K.evolution turbo type D』では，二駒関係の一種である KP を改変した特徴量を採用している．KP が「自玉または敵玉と，玉以外の駒 1 つとの位置関係」を表す特徴量であるのに対して，敵玉を含む位置関係を使用せず，自玉を含む位置関係だけを使用する．以下では，この特徴量を「HalfKP」と呼ぶことにする．

HalfKP 特徴量は，前項で説明した，手番側と非手番側の特徴ベクトルを両方とも使用する手法と組み合わせるための特徴量である．HalfKP 特徴量を使用すると，式 (17) において， $\boldsymbol{v}^{(p,\mathrm{active}(p))}$ は「手番側から見た，手番側の玉に関する KP」に相当する情報を持ち，$\boldsymbol{v}^{(p,\mathrm{opponent}(p))}$ は「非手番側から見た，非手番側の玉に関する KP」に相当する情報を持つことになる．従って，特徴量そのものには敵玉の位置に関する情報を含まないが，手番側から見た特徴量と，非手番側から見た特徴量の両方を使用することによって，局面全体の情報を表現することができる．通常の KP を特徴量として使用する場合と比較して，同等の情報を表現するために必要な計算コストが小さくなることが利点である．

2.8 整数 SIMD 演算

SIMD (Single Instruction Multiple Data) 演算は，同一の処理を複数のデータに同時に適用する演算である．NNUE 評価関数は，各部分において整数の SIMD 演算を活用して高速に計算できるように設計されている．

特徴ベクトルのアフィン変換は，差分計算を行う場合も行わない場合も，ベクトルの加減算で求められる．重み行列 $\boldsymbol{W}_1$ を 16-bit の整数に変換して保持し， 16-bit の符号付き整数ベクトルの加減算を行う命令（AVX2 では `VPADDW`, `VPSUBW`）で計算する．

特徴ベクトル以外のアフィン変換では，直前の層のアクティベーション $\boldsymbol{a}_{l-1}$ と重み行列 $\boldsymbol{W}_l$ を 8-bit で表現し， 8-bit 整数ベクトル 2 つの積和を求める命令（`VPMADDUBSW`）などを用いて内積を計算する．

活性化関数は，整数のベクトルのビット幅を変換する命令（`VPACKSSDW`, `VPACKSSWB`）と， 8-bit 整数ベクトル 2 つの要素ごとの最大値を取る命令（`VPMAXSB`）などによって計算する．

表 1 『the end of genesis T.N.K.evolution turbo type D』の評価関数で使用する重み行列のサイズ．

	列数	行数
$\boldsymbol{W}_1$	125388	256
$\boldsymbol{W}_2$	512	32
$\boldsymbol{W}_3$	32	32
$\boldsymbol{W}_4$	32	1

2.9 モデルサイズ

表 1 に，『the end of genesis T.N.K.evolution turbo type D』の評価関数で使用する重み行列のサイズを示す．列数と行数が，それぞれアフィン変換の入力と出力の次元数に対応する．このモデルサイズは，単位時間あたりに読める局面数が，評価関数に三駒関係を使用した場合と同等になるように設定したものである．

全パラメータ数の大半を，特徴ベクトルのアフィン変換に使う重み行列 $\boldsymbol{W}_1$ が占めている．この部分は差分計算を行うため，実際の計算コストは大きくない．$\boldsymbol{W}_2, \boldsymbol{W}_3, \boldsymbol{W}_4$ を使うアフィン変換は差分計算ができないが，パラメータ数を比較的少なく設定し， 8-bit 整数の SIMD 演算によって高速な計算を可能にしている．

3. おわりに

本稿では，ニューラルネットワークを用いた将棋の評価関数を提案し， CPU で高速に動作させるための設計と高速化技術について述べた．本技術による評価関数を搭載する初めての将棋ソフト『the end of genesis T.N.K.evolution turbo type D』は，第 28 回世界コンピュータ将棋選手権 [1] に参加を予定している．

また，本技術を実装したソースコードを後日公開する予定である．オープンソースの将棋ソフト『やねうら王』[8] をベースとして C++ で実装したもので，本稿で述べた内容に加えて，以下のような特徴がある．

- クラステンプレートを利用した，特徴量やネットワーク構造のカスタマイズ性
- コンパイル時計算やコンパイラによる最適化を活用した高速な実装
- 三駒関係の学習で用いられる手法と同様の，
  - 静止探索と組み合わせた学習
  - 特徴量の分解による学習時のパラメータ共有

特徴量やネットワーク構造は，『the end of genesis T.N.K.evolution turbo type D』で採用したものが最適

であるとは限らない．より優れた構造の検討は今後の課題であり，ソースコードを拡張して利用される他の開発者にも期待したい．

本技術がコンピュータ将棋の更なる発展に繋がれば幸いである．

参考文献

[1] 第 28 回世界コンピュータ将棋選手権. http://www2.computer-shogi.org/wcsc28/.

[2] 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. ゲームプログラミングワークショップ 2003 論文集, pp. 14–21, 2003.

[3] Kunihito Hoki and Tomoyuki Kaneko. Large-scale optimization for evaluation functions with minimax search. *Journal of Artificial Intelligence Research*, Vol. 49, No. 1, pp. 527–568, 2014.

[4] 金澤裕治. NineDayFever アピール文書. http://www.computer-shogi.org/wcsc24/appeal/NineDayFever/NDF.txt, 2014.

[5] 竹内章. コンピュータ将棋における大局観の実現を目指して. 人工知能学会誌, Vol. 27, No. 4, pp. 443–448, 2012.

[6] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, Vol. 529, No. 7587, pp. 484–489, 2016.

[7] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. arXiv:1712.01815v1 [cs.AI], 2017.

[8] 磯崎元洋. やねうら王オープンソースプロジェクト. http://yaneuraou.yaneu.com/yaneuraou_mini/.

チーム Barrel house のアピール文書@第 28 回世界コンピュータ将棋選手権

Barrel house とは岡山の駅前にあるビアバーです．職場や社会人勉強会などで仲間を求めさすらって行きついたところで一人目の仲間を見つけたのと，マスターに許可頂いたので名前をお借りしました．その後，メンバーが増えて当初の方向性とは全く違って来ましたが，まぁ一度出した名前を変更するのもアレなのでそのまま行きます．

プログラム名：Hefeweizen
ドイツ南部の酵母入りビール．濁った白ビールってのが日本で通る表現かと．命名経緯は上記の通りです．

チームの特徴．
メンバーが全員初参加で冷たい白ビールのようなフレッシュなチームです．
メンバー間の擦り合わせも適当ですが昨年秋の電王トーナメント経験者が 2 名おり，多少は実績がありますので乞うご期待という感じです．電子工作が得意なメンバーや家電販売が得意なメンバーもおります．結成が遅れ現在手分けしている段階ですので確定事項は少ないのですが，やねうら王の探索部とオリジナルの局面評価関数と分岐先読みクラスタを使用する予定です．深層学習で作った評価関数や魔改造した技巧 2 などがあるのですが，本番で使われるかどうかはある程度ツールが揃って強さを計測してから決定する予定です．
疎結合なクラスタ構成なのでパーツ交換で対戦ごとに構成を変える可能性もあります．

CSA 使用可能ライブラリ使用表明．
メンバーの意思統一が図れておりませんので多めに入れておきます．また，試行錯誤ツールで色々使わせて頂いております．
Apery，やねうら王，tanuki-，Qhapaq，elmo，技巧，python-shogi，人造棋士 18 号
（作者本人は入れなくていい？）

使用マシン．
普通のノートパソコンに加えて，クラウドの力をお借りする予定です．ベンチマーク等も未だなので具体的なことは全く決まっていません．

4 月 13 日追記
ライブラリの選定理由を加筆修正しろとのことですので，追加します。

ライブラリ選定理由
python-shogi：python を用いた棋譜および局面の管理，sfen 文字列の展開など
Apery：評価関数作成のための教師データ作成に利用。
やねうら王：探索部が高速なため主に探索部の利用。定跡部作成時にも利用。
tanuki-：互換性があり強いため評価関数の作成及び仮想敵として勝率計算に利用。
Qhapaq：互換性があり強いため評価関数の作成及び仮想敵として勝率計算に利用。
Elmo：互換性があり強いため評価関数の作成及び仮想敵として勝率計算に利用。
技巧：評価関数の作成及び仮想敵として勝率計算に利用。探索部のオーダリングにも利用。
人造棋士 18 号：独自のルーチンを用いた評価関数の作成に利用。

クラスタ形式でのテストをまだ行っていないため本戦での運用で若干変更があるかもしれません。

第28回世界コンピュータ将棋選手権
Aperyアピール文書
2018年3月31日 平岡 拓也、杉田 歩

・Aperyとは
- 読み方は「えいぷりー」です。
- GPL v3ライセンスでオープンソースで開発しています。
- ソースコードや実行ファイルは以下で公開しています。
- http://hiraokatakuya.github.io/apery/
- CSAライブラリとして登録しているので、誰でもAperyを改造してご自身のソフトとして出場出来ます。

・作者紹介
- 平岡 拓也 (ひらおか たくや)
  趣味で作っています。
  Mail　: hiraoka64@gmail.com
  Twitter: @HiraokaTakuya
- 杉田 歩 (すぎた あゆむ)

・実績
- 第22回世界コンピュータ将棋選手権 22位 (2次予選敗退)
- 第23回世界コンピュータ将棋選手権　9位 (2次予選敗退)
  決勝進出に一歩足りず。
  新人賞に一歩足りず。
- 第1回将棋電王トーナメント 6位 (5位決定戦敗退)
  第3回電王戦出場に一歩足りず。
- 第24回世界コンピュータ将棋選手権 優勝
- 第2回将棋電王トーナメント 5位
- 将棋電王戦FINAL 斎藤慎太郎五段(段位は当時)に完敗
- 第25回世界コンピュータ将棋選手権 4位
- 第3回将棋電王トーナメント 3位
  出場ソフト名は「大樹の枝」。
  名前はヤフオク！の「みんなのチャリティー」で命名権をオークション(全額チャリティー)で販売して落札者の方に決めて頂きました。
- 第26回世界コンピュータ将棋選手権 4位
- 第4回将棋電王トーナメント 2位
  出場ソフト名は「浮かむ瀬」。
  名前はヤフオク！の「みんなのチャリティー」で命名権をオークション(全額チャリティー)で販売して落札者の方に決めて頂きました。
- 第26回世界コンピュータ将棋選手権 12位
- 第5回将棋電王トーナメント ベスト12

・技術的特徴
- 全体的にチェスソフトのStockfishの設計を取り入れており、評価関数は3駒関係です。利きのデータなどは持っておらず、シンプルな構成になっています。
- 評価関数で手番も評価しています。
- 評価関数の教師データ生成を皆さんにお願いするシステムを今年も使っています。
  今回は Linux 限定になっています。
  ご協力ありがとうございます！
  https://github.com/HiraokaTakuya/apery-machine-learning

・昨年との違い
- 昨年の選手権で elmo が採用していた、評価関数の学習データ生成時の対局で、
  勝敗を記録し、学習にその局面と後の勝敗を考慮する方式を取り入れました。
- 学習部を書き直した過程でバグが取れたのか、WCSC27 の elmo の評価関数より精度が上がり、
  半年前の第5回将棋電王トーナメント時には、オープンソースの中では最も評価関数の性能が高かったようです。
- 半年前からの変更点は、学習データ生成時の探索深さを8から10に変更した点です。
  とても計算量が多いので、半年前は出来ませんでしたが、やることにしました。
  これは DeepMind の AlphaZero が AlphaGo と同様の方法で将棋でも既存のソフトより強くなったとい

う発表を受けて、
　単に学習に使った計算量が桁違いに大きいからではないかと考えたからです。
　既存手法でももう少し計算量を増やして学習すれば、更に強くなるのではないかと考えました。
　具体的には、探索深さを 10 で学習し、その後 12 でもう一度学習する計画でした。
　学習データ生成を誰でも出来るシステムなども作りましたが、時間的に探索深さ 12 は実験出来そうにないです。
　まだ実験中で、探索深さ 10 でのデータ量が 8 の時よりも少し少なかったりで、結論を出すには早いですが、
　探索深さ 10 の時点で伸びがほとんど無いかも知れません。
　本番までにもう少し実験しようかと思います。

第28回世界コンピュータ将棋選手権

PAL
アピール文書

山口 祐

PALの概要

- pal /ﾊﾟﾙ/【名】<話> 仲間、友だち、仲よし
  (例) a pen ー
- **ショーギはともだち** こわくないよ!

開発者

- 山口 祐 @ymg_aq
- 囲碁もやってます
  ➡️ https://github.com/ymgaq/AQ

PALの特徴

- Deep Learningを**使わない**
- 0ベースの評価関数の学習
- 強化学習と **敵対的学習** のハイブリッド
- クラウドサーバにおける**高回転並列学習**（N連ガチャ）
- 並列探索の最適化

使用ライブラリ

- やねうら王 v4.80　（改変しやすそうだったので）

コンピュータ将棋ソフト
「大合神クジラちゃん」

第 28 回世界コンピュータ将棋選手権　アピール文書

2018/04/11　鈴木雅博、大賀一貴

1. 大合神クジラちゃんの紹介

マスタ／スレーブの構成を取り、マスタがスレーブを管理しながらクラスタリング探索を行います。スレーブを動かすのは主にニコニコ生放送や YouTube 視聴者のパソコンを考えています。前回の選手権ではパソコン４００台以上のクラスタで動かしました。

クラスタの基本的な手法は GPS 将棋を参考にしています。マスタが各指し手をスレーブに展開することでより読みを深くしています。また、冗長性を持たせるため同じ手を 2 つのスレーブに探索させています。

2. 独自の工夫

独自の工夫として、探索を探索ツリーの末端ノードだけでなくすべてのノードで行うことで、スレーブの性能差を気にすることなく安定した棋力向上を可能にしました。　過去のシステムでは末端でないノードでは探索させず、末端ノードのみに指し手を展開し、展開できなかった手はそれ以外の手を探索する「その他ノード」に探索させるようにしていました。しかしながらその手法では、ある指し手とそれ以外の指し手の探索量が大幅に違った場合、なにが最善手であるか判断するのが極めて難しいという問題がありました。具体的には、評価値が 1000 点・探索深さ 14 という手と、評価値が 900 点・探索深さ 20 という手があった場合、どちらが良い手なのか判断できないという問題がありました。

現在のシステムでは、末端以外の親ノードでも探索を行うようにしこの不安定性を少し減らすことに成功しました。例えば上記の例であれば、末端ノードの探索結果（指し手 A：評価値が 1000 点・探索深さ 14、指し手 B：評価値が 900 点・探索深さ 20）に加え、親ノードの探索結果が加わります。例えばその結果が評価値 950・探索深さ 18 で指し手が A だったとすると、探索深さが深くなった場合でも指し手 A が有力ということがわかります。

またこのシステムはビンゴマスターさんのアドバイスをもとに作られたものなので、ビンゴマスターさんに深く感謝申し上げます。

ディープラーニングによる指し手探索もやろうと考えていますが、現状（4/11）全く手つかずです。もし間に合いそうであればクラスタの指し手の初期探索部分を DL で行いたいと考えています。

3. その他

- 探索部分に YaneuraOu、評価関数は既存のものを追加学習したバージョンを使用しております。
- 非公開協力者として、まふさんに定跡ファイルを作って頂く予定です。
- ディープラーニングによる指し手探索もやろうと考えていますが、現状（4/11）全く手つかずです。もし間に合いそうであれば指し手の初期探索部分を DL で行いたいと考えています。

4. ライブラリの選定理由について

- YaneuraOu
  - クラスタのスレーブ部分の基礎として使用。非常に強く探索部分も優秀なため使わせて頂いています。
- Apery
  - クラスタのマスター部分の将棋ライブラリとして使用。探索は行わず合法手の確認や指し手生成に使っています。YaneuraOu に変えてもいいのですが、昔から使っているので今も使用しています。
- 人造棋士 18 号
  - 関連ツールを使用させて頂く可能性があるため登録しています。
- dlshogi
  - マスターの初期探索部分をディープラーニングで行おうかと考えているので、念のため登録しております。ただし現在は全く手がついておりません。

5. 過去の戦績

- 第 23 回コンピュータ将棋選手権 27 位（独創賞を獲得）
- 第 24 回コンピュータ将棋選手権 16 位
- 第 25 回コンピュータ将棋選手権 19 位
- 第 27 回コンピュータ将棋選手権 4 位

6. 完全オリジナルのＧＵＩもあります。

- コンピュータ将棋を動かすための GUI も自作しています。
- クジラちゃんを擬人化し、GUI 内に取り込みました。
- URL: http://garnet-alice.net/programs/whalewatcher/
- ニコニコミュニティ： http://com.nicovideo.jp/community/co516151

名人コブラアピール文書

生粋のライブラリ勢

過去の経験から、ポンコツプログラマな自分がプログラムを改造すると大幅な低速化を招くことがわかりました。ですから、今回も探索部や評価部には手を付けないことにしました。低速化をしても直接的には読みのスピードに影響の無い、学習部だけに手を加えたいと思いいます。

使用ライブラリ

- Apery
  - 評価関数だけならば、トップを走っているといわれているので、学習部を改造して使用させていただきたいと思います。キメラの素として、評価関数も使用させていただきます。
- やねうら王
  - 現在探索部は最強だといわれているので、探索部を使用させていただきたいと思います。また学習部も改造して使用させていただきたいと思います。キメラの素として、評価関数も使用させていただきます。
- elmo、人造棋士 18 号
  - Apery-やねうら王系統の評価関数を使用しているため、キメラの素として、評価関数を利用させていただきます。

評価関数のキメラ化にこだわります

評価関数をブレンドすると強くなるらしいので、少なくとも 20 個の評価関数を作って、それを混ぜたいと思います。お手軽だし、「アンサンブル学習」という言葉やコンセプトがなぜか大好きなのです。

キメラ素材の調達方法

1. ライブラリの評価関数

   ある程度実績のある評価関数を使用させていただきたいと思います。

2. ゼロから新たに学習させた評価関数

   同じものを混ぜても、キメラ化の効果はありません。バリエーションが必要なので、次の方法でバリエーションを持たせたいと思います。

   - 教師局面の復元抽出
   - 特徴の部分的使用（ランダム部分空間）

教師局面と評価値

キメラ素材となる評価関数の学習には以下の教師局面と評価値を使用します。

1. Apery を使用して生成したもの
2. やねうら王作者の磯崎氏が配布しているもの

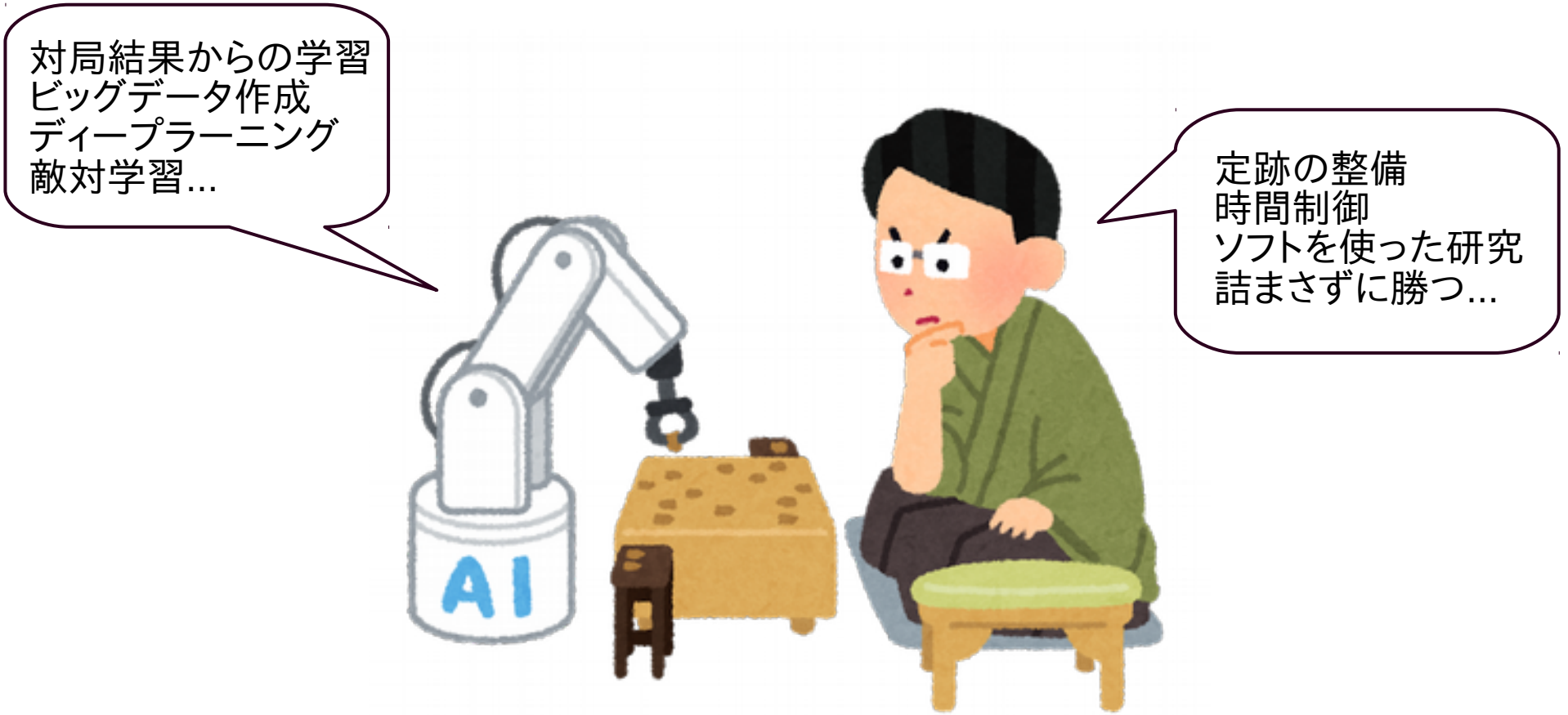
最後に

ライブラリ勢なのに今まであまりライブラリを使いこなせていなかった感じがします。今回はあまり余計な改造はせず、ライブラリを使いこなして、良い評価関数を作ることに重点を置きたいと思います。

妖怪惑星Qhapaqのアピール文

Ryoto Sawada, Yuki Ito and Toshihiro Shirakawa

ITに強い将棋部 (updated 2018/04/08)



Who is Qhapaq ?

カパック（ケチュア語で「偉大なもの」を指す形容詞）と読みます。本作が数々の巨人の肩に立った作品であることを示しています。

数理解析を駆使したご家庭レベルのPCで行える強化学習や、高速化を売りにしています

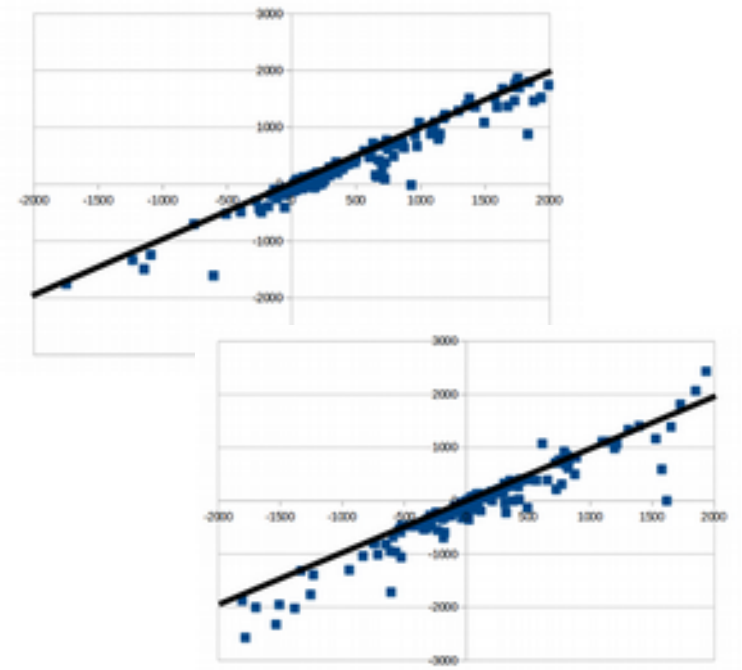
主な実績：
・第27回世界コンピュータ将棋選手権10位
・第五回将棋電王トーナメント：5位入賞
・現在最強の公開関数Apery-Qhapaq関数
・電王トーナメント累計23戦中18回後手

チーム名：ITに強い将棋部

Ryoto Sawada : 某企業 and 大学の研究者。専門は量子物理
Yuki Ito : Ubuntu帽を被った野生の物理学徒。高速化のプロ
Toshihiro Shirakawa : パズル世界の怪人。趣味は電子ゴルフ

将棋フィーバー(ただしAI)

AIを使って将棋の可能性を広げたい！



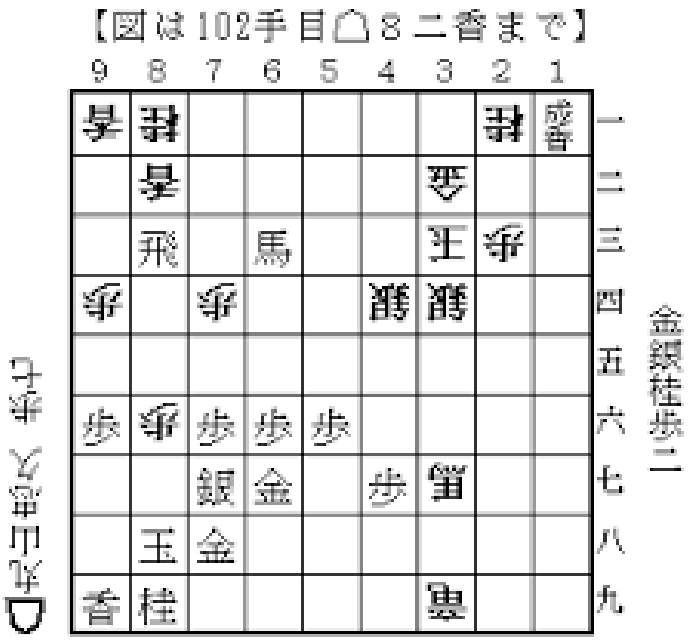
将棋界：
- 藤井フィーバー
- 羽生永世七冠誕生
- ひふみん超おもしろい

AI将棋界：
- Ponanza引退
- オープンソース勢の大躍進
- AIブームに便乗した手法の進歩

評価値推移から見る棋風解析
藤井 vs 羽生は藤井勝率65%ぐらいらしい(Qhapaq曰く)

1年前のチャンピオンとのレート差が300-400。角落ちのレート差が600ぐらいと言われているので、2年で角落ちを埋める程度の成長率

Qhapaqの挑戦：まず強くする



今の流行は強化学習

- 人間やソフトの棋譜を使った学習
- 勝った局面は良い、負けた局面は悪い
- ソフトの評価値と勝敗を合議(elmo絞り)
- やねうら王などのライブラリで個人でも再現可能

今後の課題：

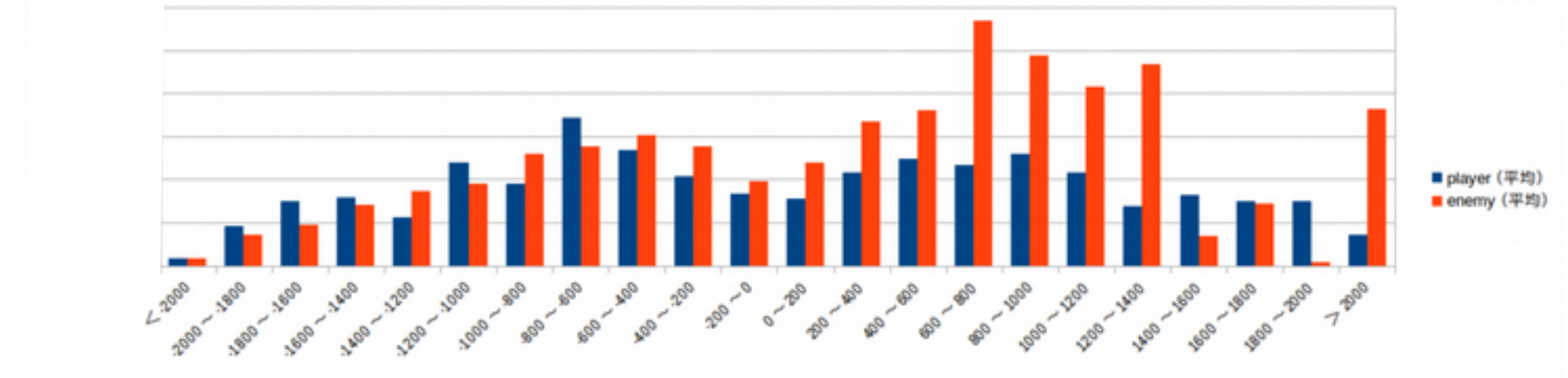
- 普通に棋譜を作ると滅茶苦茶計算資源がかかる
- 教師の数、読みの深さ双方が必要
- 人間ならすぐ駄目と解ることでもコストが高い
- 評価関数は線型のままで良いのか

Qhapaqの工夫：

- floodgateなどで得られる深い読みの局面を利用（教師精度はNo1）
- 教師データが少ないのでその局面から浅い読みで作られた局面も教師にする
- その教師の勝敗や評価値は他データから類推する
- 今で言うVirtual Adversarial Trainingに近い
- KPPTを強くする以外にも新規評価関数開拓をやってる
- 高速化もやってる（チームメンバーのItoさんが）

Qhapaqの挑戦2：将棋を面白くする

棋力解析ソフトELQ



上の図は藤井六段と対局相手のソフト視点での悪手率。
横軸は評価値（左が不利な局面、右が有利な局面）

- 藤井六段、対局相手の双方とも、-600～600ぐらいのイーブンな局面が一番ミスが出やすい
- 藤井六段は有利な局面でのミスが少ない。詰将棋万歳

このデータを用いて対局者の強さを予想することも可能。藤井六段は
今後羽生永世七冠を越えて行くとQhapaqは予想していますが果たして...

利用ライブラリとその選定理由

・Apery
評価関数の初期値に使っています。また、学習部には
Aperyの学習部を改造したものを用いています

・やねうら王
探索部、学習部として使っています

・技巧
進行度などの取り扱いや、ssh通信部は技巧のコードを
参考にしています。ゼロから書き直しているとはいえ、コピペ
部分も多いのでライブラリ申請しておきます

最後にひとこと

Qhapaq強いですよ!

将棋ブームをエンジョイしてるそこの人、コンピュータ将棋も見てくれると嬉しいです!

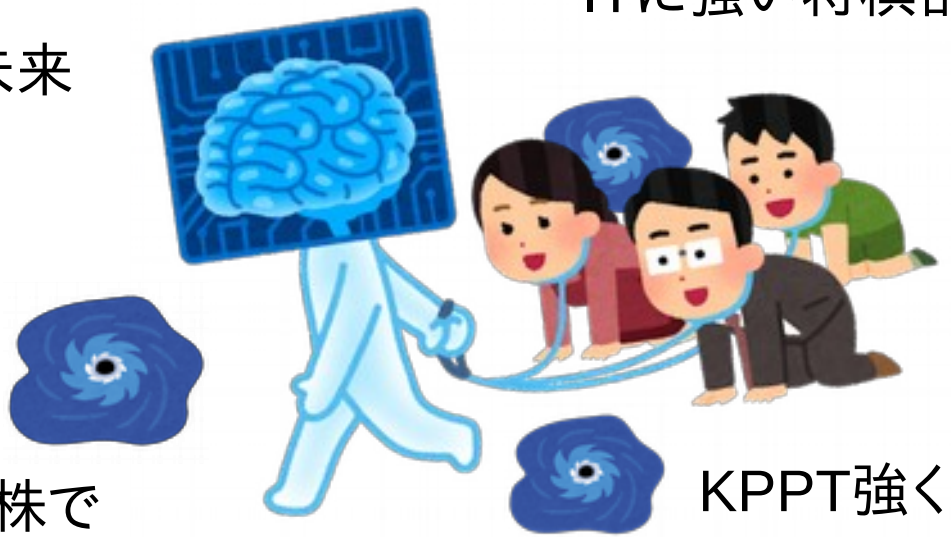
再放送

妖怪惑星Qhapaqのアピール文

Ryoto Sawada, Yuki Ito and Toshihiro Shirakawa

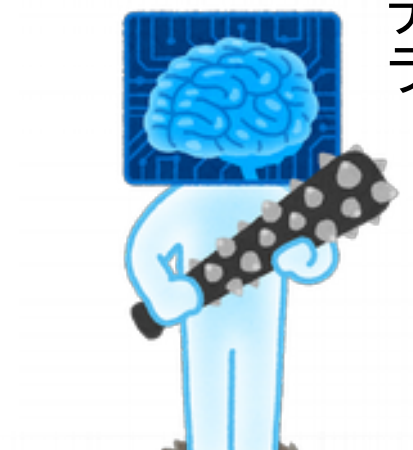
AIが導く
明るい未来

ITに強い将棋部



KPPT強くするぞ

ディープ
ラーニング



ハイテク株で
X万円を溶かした
Qhapaq

GPU

小市民

Who is Qhapaq ?

カパック（ケチュア語で「偉大なもの」を指す形容詞）と読みます。本作が数々の巨人の肩に立った作品であることを示していますが後述するやらかしは巨人たちの責任ではありません。

主なやらかし:
・被り物を被ってニコニコ動画出演
・振り駒の結果に中指を立てる
・対戦カードを正しく組んでもらえない
・通信器具を会場に持ち込み忘れる
・電王トーナメントでのサイリウム配布テロ
・なんとかchに経歴を晒される

チーム名:ITに強い将棋部

Ryoto Sawada : wikipediaに名前がある。1日の食費は300円ちょい
Yuki Ito : 異議あり民の一人。ラーメン屋で外人に絡まれて困惑してた
Toshihiro Shirakawa : 物書き時々ﾕｰﾁｭｰﾊﾞｰ。服装がﾌｧﾝﾀｽﾃｨｯｸ

将棋フィーバー(ただし身内)

AIを使って開発者のキャリアを広げたい！

- bing+カタカナだと出てこない
- googleだと将棋をサジェストされる
- Qhapaqで検索すると出てくる
- カタカナ検索の結果は諸事情で出せない

将棋界：
- 三浦九段の冤罪事件
- 加藤 One hundred twenty three
- 某電王による大量献金

妖怪惑星：
- アカデミアから(強制)引退
- AIブームに便乗(ただしAI株で失敗)
- 履歴書に書けるネタを稼いでｷｬﾘｱｱｯﾌﾟ

最近AIができる物理屋ではなく、物理ができるAI屋と思われているフシが有る。

Qhapaqの挑戦：人の失敗を吊るし上げる

elmo絞りの此処が駄目：

- 短い持ち時間だと強くならないことが多い
- 一度しか出てないKPPTの値がelmo絞りだと大きくなりすぎる問題
- 教師局面110億とか狂気の沙汰
- ニューラルネット系の学習だと超強力

評価関数合成の此処が駄目：

- 合議制では強くなるような関数でも強くならない(ことがある)
- 強くなる関数では合議より強くなる
- どの成分の変化がレート変更に結びつくかを分析する必要あり

Qhapaqの此処が駄目：

- こうご期待の一言でアピール文を片付けられず、大会前に役立つ情報を漏らしてしまう

Qhapaqの挑戦2
ソフト開発者 ＝ 秀才の構図を破壊する

■参加プログラム情報

チーム名：

振り飛車!飛車!飛車!飛車ぅぅううぅわぁあああああああああああああああああああああああああん!!! あぁあああああ…ああ…あっあっー!あぁあああああああ!!!飛車飛車飛車ぅううぁわぁあああああ!!! あぁクンカクンカ!クンカクンカ!スーハースーハー!スーハースーハー!いい匂いだなぁ…くんくん んはぁっ!48に居る飛車たんの裏面の桃色ブロンドの文字をクンカクンカしたいお!クンカクンカ!あぁあ!! 間違えた!モフモフしたいお!モフモフ!モフモフ!龍の文字モフモフ!カリカリモフモフ…きゅんきゅんきゅい!!りゅうおうのおしごと7巻絶賛発売中だよぅ!!あぁぁああ…ああああ…あっあぁあああああ!!ふぁぁあああんんっ!! sdt未だ終わってなくて良かったね飛車たん!あぁあああああ!かわいい!飛車たん!かわいい!あっあぁぁああ! アニメも評価されてて嬉し…いやぁあああああああ!!!にゃあああああああああん!!ぎゃあああああああ!! ぐあああああああああああああ!!!将棋なんて現実じゃない!!!!あ…小説もアニメもよく考えたら… 振り飛車が勝ち越す未来は現実じゃない?にゃあああああああああああああああん!!うぁあああああああああああ!! そんなぁあああああああ!!いやぁぁぁああああああああああ!!はぁあああああああん!!48飛車ぁあああああ!! この!ちきしょー!やめてやる!!振り飛車なんかやめ…て…え!?見…てる?角交換四間飛車を指しこなす本の飛車ちゃんが僕を見てる? 四筋の飛車ちゃんが僕を見てるぞ!飛車ちゃんが僕を見てるぞ!駒台の角ちゃんも僕を見てるぞ!! アニメのあいちゃんが僕に話しかけてるぞ!!!よかった…世の中まだまだ捨てたモンじゃないんだねっ! いやっほぉおおおおおおお!!!僕には将棋ちゃんがいる!!やったよケティ!!ひとりでできるもん!!! あ、コミックのひなたちゃあああああああああああああああん!!いやぁあああああああああああああああ!!!! あっあんああっあんあアン様ぁあ!!セ、セイバー!!シャナぁあああああああ!!!ヴィルヘルミナぁあああ!! ううっうぅうう!!俺の想いよ将棋盤へ届け!!readyok! と言った感じでITに強い将棋部は常に最強を求め将棋ソフト開発に余念がない。しかし、開発の疲れからか不幸にも疲れからか、不幸にも黒塗りの高級車に追突してしまう。後輩をかばいすべての責任を負った三浦に対し、車の主、暴力団員谷岡に言い渡された示談の条件とは…。

WCSCのサイトが前時代的なcgiであり
チーム名の文字数制限が抜けていたので
チーム名をルイズコピペにしてみた
（360RTぐらいされた）

たぬき開発者と共謀して電王トーナメントの
ライブに自腹でサイリウムを持ち込んだ

肩を借りた巨人とその選定理由

・既存定跡を組み込んだ量子アニーリング
定跡作りの際に使っています。複数の既存定跡にノイズ（具体的には一定確率で定跡を無視する）を加えたものと自己対局を行い、勝った棋譜を収集することを繰り返すことで、未知の定跡に対してロバストな定跡を作っています

・妖怪惑星クラリス
今回のソフト名の元ネタです。このゲームが末永く発展することを願って付けました

・Lostorage conflated WIXOSS
sdt5でのソフト名の元ネタです。四期の放送（当時未確定）を願って付けました。願いが叶ったわけですが、これはもしや...

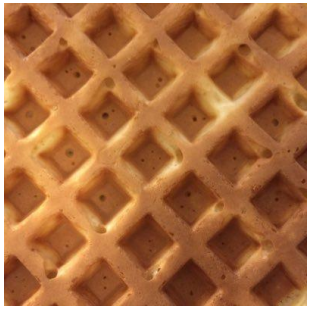
最後にひとこと

妖怪惑星クラリスが唐突にサービス終了しました

ｳｷﾞｬｱｱｱｱｱｱ


HoneyWaffle
第28回世界コンピュータ将棋選手権　アピール文書
開発者　渡辺 光彦

開発者



氏名: 渡辺 光彦

職業: プログラマー

棋力: 将棋ウォーズで3級、ぴよ将棋でR700-800程度の振り飛車党

Twitter: @shiroi_gohanP (https://twitter.com/shiroi_gohanP)

ニコ生の電王戦をきっかけにコンピュータ将棋を始める。

将棋連盟Liveやニコニコ放送、AbemaTVの将棋中継が好き。

HoneyWaffle (ハニーワッフル) 名前の由来

・四角いワッフルは将棋盤と似ている

・ゆるふわスイーツ的なスナック感覚の軽さを表現

　元々タブレット向けに開発していたので物理的に軽いこと、振り飛車の軽い捌きができるようになるといいなという想いから命名しました。

※表紙やTwitterアイコンのワッフルはうちで焼いたものを使用しています。

以下のリンク先で出せるものは公開しています。使い方がおかしいのはいつものこと。

https://github.com/32hiko

戦績

2016年、Go言語でオリジナル開発版

・2016年5月 第26回世界コンピュータ将棋選手権 出場　一次予選2勝5敗

・2016年10月 第4回将棋電王トーナメント 出場　予選リーグ3勝5敗

2017年、やねうら王ライブラリ使用

・2017年5月 第27回世界コンピュータ将棋選手権 出場　決勝リーグ7位

・2017年11月 第5回将棋電王トーナメント 出場　決勝トーナメント初戦敗退

本題に入る前にポエム①

今回のコンセプトの前に、将棋とは何かをもういちど考える。将棋とは？

> 二人が「盤」の上で交互に「駒」を動かします。そして相手の「玉将（ぎょくしょう）」という駒を先に捕獲したほうが勝ちとなります。

（日本将棋連盟サイトから引用　https://www.shogi.or.jp/knowledge/about/ ）

→先に捕獲するとは、先に攻めるということ？

→先に攻めるのが有利ということ？

→そう考えるのが自然か。

本題に入る前にポエム②

ここ最近の将棋（コンピュータ将棋や、それに影響を受けたプロの将棋）は、確かに先に攻めようとする傾向がさらに強くなったかも。

・玉は1手2手動かすだけで、金銀もあまり玉に近づけずバランス重視（角換わりとか横歩取りとか）

・じっくりした矢倉が減った

・早く攻めるために桂馬も早く跳ねる

今現在の3駒関係の学習も、要は「早く勝ちやすい形を高く評価する」もの （現局面の評価値を、何手か先の局面の評価値に近づくよう学習する。勝った局面はより高く評価するよう学習する。）

本題に入る前にポエム③

「早く攻める」のを重視する前提なら、なるほど振り飛車は不利だと言わざるを得ない。少なくとも飛車を振る手で1手かかってるのは事実。工夫しないで学習回すと、飛車振らなくなっていくのも納得。

ただ、逆に、将棋の勝敗そのもので不利なわけではないよねと。純粋に手数がのびたとしても、最後に相手よりも1手早いだけでいいから。

先に攻めるのは先に詰ます可能性もあるけど、先にカウンターを食らう、先に攻めが切れるという恐れもあるよねと。

相掛かりで5手目に▲２四歩から無理やり攻めようとすると、先に歩を手にした後手が有利になる変化。他の戦型でも極限まで攻めを早くしていったら、それと似たような局面ばかりになるかもしれない。

本題に入る前にポエム④

現在の潮流（居飛車が多い）：最速の攻めを目指す。攻撃が最大の防御。

→相居飛車でお互いに最速の攻めを目指すのは、戦型にこだわらない陣営が自然と当たり前にやっている。

→ならば、それに対抗できる振り飛車のソフトはかくあるべし！

・最速の攻めが来てもギリギリ受けられる受けの力（構想、囲い）

・隙を見て豪快に攻め、鮮やかに1手違いで勝つ

定跡や変な評価関数でただ飛車を振るだけでは真の「振り飛車」ではない。1手遅れたあげくに、中途半端に囲って、それから最速の攻めを目指してもね…。この辺は割と当たり前というか基本のはずなのにどうして…。

コンセプト

・「公開されている技術を最大限に活用し、最強の振り飛車党ソフトを目指す」

実は第5回将棋電王トーナメントの時と同じですが、意味合いはポエムで述べた通りまったく違います。去年一年、飛車を振らせるのに苦労したのは確かなのですが、それが精一杯で、考えが全然足りませんでした。

一般的に形勢判断の基準は①駒の損得②駒の働き③玉の堅さ④手番と言われていますが、現状の3駒関係では、①②④の評価がとても洗練された半面、前述のとおり攻めを重視した結果③の影が薄いように思います。意識して「堅く囲ってから鋭く攻める」ことができるよう、以前までの取り組みに加えて③を重視します。

ハードはAWS EC2インスタンス1台（極力高性能なものを選択）の予定です。

構成

■開発部

ライブラリ申請したやねうら王を強引に改造（既存の3駒での評価に加え、別に堅さ評価する等）★採用理由：最高の探索、フレームワーク。去年から使用している。

AWS接続兼時間攻めモジュール（ローカルにて使用）は自作

■評価関数

ライブラリ申請したAperyのものがベース　★採用理由：最高の評価関数

教師局面生成にて評価値を意図的に改変し追加学習する等で調整　https://github.com/32hiko/HoneyWaffleSDT5

■定跡

第5回将棋電王トーナメントで使用したものをベースに、より持久戦を目指す方針で調整

最後まで読んでいただき、ありがとうございます。