DEPARTAMENTO DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Análisis de feature sets con redes neuronales NNUE para engines de ajedrez

May 30, 2024

Martín Emiliano Lombardo
mlombardo9@gmail.com

Directores

Agustín Sansone
agustinsansone7@gmail.com

Diego Fernández Slezak
dfslezak@dc.uba.ar

Facultad de Ciencias Exactas y Naturales

Universidad de Buenos Aires

Ciudad Universitaria - (Pabellón Cero + Infinito)

Intendente Güiraldes 2610 - C1428EGA

Ciudad Autónoma de Buenos Aires - Rep. Argentina

Tel/Conmutador: (+54 11) 5285-9721 / 5285-7400

https://dc.uba.ar

Contents

1 Introduction

1.1 Chess Engines

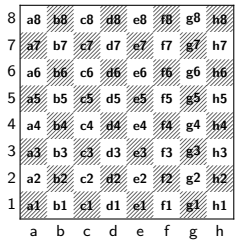
2 Feature set (board encoding)

To evaluate chess positions, we will use a neural network with an architecture explained in detail in the next chapter. In this chapter, we will build the one-dimensional input vector for such network, which can be described entirely by a feature set.

A feature set is a set built by a cartesian product of smaller sets of features, where each set extracts a different aspect of a position. Each tuple in the feature set corresponds to an element in the input vector, which will be set to 1 if the aspects captured by the tuple is present in the position, and 0 otherwise. If a tuple is present in a position, we say that the tuple is *active*.

Let's consider some basic sets of features. The following sets encode positional information about the board:

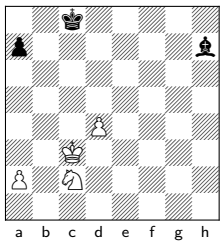
$$\begin{aligned} \textsc{File} &= \{a, b, ..., h\} \\ \textsc{Rank} &= \{1, 2, ..., 8\} \\ \textsc{Square} &= \{a1, a2, ..., h8\} \end{aligned}$$



And the following encode information about the pieces:

$$\begin{aligned} \textsc{Role} &= \{ \text{♙ Pawn, ♘ Knight, ♗ Bishop, ♖ Rook, ♕ Queen, ♔ King}\}^1 \\ \textsc{Color} &= \{\text{○ White, ● Black}\} \end{aligned}$$

Since each set has to capture some information from the position, it must be stated explicitly. For example, consider the feature set $\textsc{File}_P \times \textsc{Color}_P$ where P is *any* piece in the board, meaning that the tuples $(file, color)$ that will be active are the ones where there is at least one piece in $file$ with the color *color* (disregarding any other kind of information, like the piece's role). Another possible feature set could be $\textsc{File}_P \times \textsc{Role}_P$, with a similar interpretation. An illustration of the active features of these two feature sets for the same board is shown in Figure 2.



	Feature set	
	$\textsc{File}_P \times \textsc{Color}_P$	$\textsc{File}_P \times \textsc{Role}_P$
Active features	(a, ○), (a, ●), (c, ●), (c, ○), (d, ○), (h, ●)	(a, ♙), (c, ♔), (c, ♘), (d, ♙), (h, ♗)

Figure 1: Active features of the feature sets $\textsc{File}_P \times \textsc{Color}_P$ and $\textsc{File}_P \times \textsc{Role}_P$ for the same board

[1]The color of the pieces have no meaning in the definition. They are present for illustrative purposes.

2.1 Sum $\oplus$

The sum of two feature sets A and B, denoted by $\mathrm{A} \oplus \mathrm{B}$, is a new feature set comprised of the tuples of both sets A and B. These tuples do not interfere with each other, even if they have the same basic elements (e.g. h, 8, ♖, ●), they **must** have different interpretations. For example, given the feature sets $\textsc{File}_W$ where W is any white piece in the board and $\textsc{File}_B$ where B is any black piece in the board, the feature set $\textsc{File}_W \oplus \textsc{File}_B$ will have the basic elements $\{a, b, ..., h\}$ for both white and black pieces, but each with a different interpretation.

The sum operator is useful when we want to let the network find patterns combining information between two sets of features.

2.2 Indexing

The input to the network is a one-dimensional vector, so we need a way to map the tuples in a feature set to the elements in the input vector. The correct index for a tuple is computed using the order of the sets in the cartesian product and the size of each set, like strides in a multi-dimensional array. For this to work, each element in a set S must correspond to a number between 0 and $|S| - 1$. For example, the feature set $A \times B \times C$ has $|A| \times |B| \times |C|$ elements, and the tuple (a, b, c) is mapped to the element indexed at $a \times |B| \times |C| + b \times |C| + c$.

The same striding logic applies to feature sets built with the sum operator, recursively. [example?]

2.3 Feature sets

In this section, we will define the feature sets that will be used in the experiments. We will start with some of the most basic yet reasonable feature sets, then move to feature sets that are used by engines or were used in the past, and finally some that have not been tried, to the best of our knowledge.

2.3.1 Piece

This feature set is the most natural encoding for a chess position. There is a one-to-one mapping between pieces in the board and features:

$$\textsc{Piece} = \textsc{Square}_P \times \textsc{Role}_P \times \textsc{Color}_P$$
$$\text{for every } P \text{ piece in the board}$$

$$64 * 6 * 2 = 768 \text{ features}$$

For every position, role and color each piece could be, there is a feature. There are 16 tuples in the set that will never be active: (a8..h8, ♙, ○) and (a1..h1, ♙, ●) that correspond to the white pawns in the last rank and the black pawns in the first rank.

This is because pawns promote to another piece when they reach the opponent side of the board. Effectively, these will be dead neurons in the network, but this way we can keep the indexing straightforward. Most feature sets will have dead features, and the same logic applies.

2.3.2 Compact

This is a very compact feature set that still retains all the information of the board, meaning everything can be reconstructed by the neural network:

$$\text{Compact} = (\text{File}_P \times \text{Role}_P \times \text{Color}_P) \oplus (\text{Rank}_P \times \text{Role}_P \times \text{Color}_P)$$
$$\text{for every } P \text{ piece in the board}$$

The Compact feature set has $2 * (8 * 6 * 2) = 192$ features.

2.3.3 King-Piece

King-Piece = $\text{Square}_K \times \text{Piece}_P$ where K is the king to move and P is any *non-king* piece

$\langle side\_king\_square, piece\_square, piece\_type, piece\_color \rangle$ excl. king

$64 * 64 * 5 * 2 = 40960$ features

There are variations to this feature set, such as HalfKAv2 or notably HalfKAv2_hm that is currently the latest feature set used by Stockfish 16.1. I will not consider them in this work.

known as "KP" in the literature

if we skip the king, you may be thinking where does it get the information about the other king's side, ... blabla arquitectura Half

2.3.4 Piece+Moves

This feature set comes up from seeing the patterns recognized by the Piece feature set in section 5.5.5. When we observe... attack patterns...: P..

HalfP $\oplus \langle move\_from, move\_to \rangle$

$768 + 64 * 64 = 4864$ features

Not friendly to efficiently update the network. It is almost always better to do a full refresh on eval.

2.3.5 Half-Relative(H|V|HV)King-Piece

$\langle side\_king\_file - piece\_file + 7, side\_king\_rank - piece\_rank + 7, piece\_type, piece\_color \rangle$ excl. king

$15 * 15 * 5 * 2 = 2250$ features (for HV)

only H or only V have $8 * 15 * 5 * 2 = 1200$ features

2.3.6 Half-Top(PP)

Statistical feature set, blabla, wasted features blabla

2.4 Summary

Feature set	Tuple	# features
Piece	$\text{Square}_P \times \text{Role}_P \times \text{Color}_P$	768
Compact	asd	192
Piece+Moves	asd	4864
King-Piece	asd	40,960
RelativeHV-King-Piece	asd	2250
TopPP	asd	64

Table 1: Comparison of feature sets

3 Efficiently updatable neural networks

NNUE (ƎUИИ Efficiently updatable neural network) is a neural network architecture that allows for very fast inferences. It was invented for Shogi by Yu Nasu in 2018.

In essence, NNUEs "Neural Network Update Efficient" are just regular neural networks that allow for really fast inferences.

... Most of the information described here can be found in Stockfish's documentation about NNUEs [1].

It is important to combine this with aggresive quantization techniques.

3.1 Architecture

arquitectura half, dos capas

3.2 Efficient updates

pesada al principio y liviana al final, acumular filas de la primera capa en domove, undomove

3.3 Stockfish quantization scheme

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. $\sin^2(\alpha) + \cos^2(\beta) = 1$. If you read this text, you will get no information $E = mc^2$. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. $\sqrt[n]{a} \cdot \sqrt[n]{b} = \sqrt[n]{ab}$. This text should contain all letters of the alphabet and it should be written in of the original language. $\frac{\sqrt[n]{a}}{\sqrt[n]{b}} = \sqrt[n]{\frac{a}{b}}$. There is no need for special content, but the length of words should match the language. $a\sqrt[n]{b} = \sqrt[n]{a^n b}$. Hello, here is some text without a meaning. $\mathrm{d}\Omega = \sin\vartheta \mathrm{d}\vartheta \mathrm{d}\varphi$. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. $\sin^2(\alpha) + \cos^2(\beta) = 1$. This text should contain all letters of the alphabet and it should be written in of the original language $E = mc^2$. There is no need for special content, but the length of words should match the language. $\sqrt[n]{a} \cdot \sqrt[n]{b} = \sqrt[n]{ab}$. Hello, here is some text without a meaning. $\frac{\sqrt[n]{a}}{\sqrt[n]{b}} = \sqrt[n]{\frac{a}{b}}$. This text should show what a printed text will look like at this place. $a\sqrt[n]{b} = \sqrt[n]{a^n b}$. If you read this text, you will get no information. $\mathrm{d}\Omega = \sin\vartheta \mathrm{d}\vartheta \mathrm{d}\varphi$. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression

of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language. $\sin^2(\alpha) + \cos^2(\beta) = 1$. Hello, here is some text without a meaning $E = mc^2$. This text should show what a printed text will look like at this place. $\sqrt[n]{a} \cdot \sqrt[n]{b} = \sqrt[n]{ab}$. If you read this text, you will get no information. $\frac{\sqrt[n]{a}}{\sqrt[n]{b}} = \sqrt[n]{\frac{a}{b}}$. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. $a\sqrt[n]{b} = \sqrt[n]{a^n b}$. This text should contain all letters of the alphabet and it should be written in of the original language. $\mathrm{d}\Omega = \sin\vartheta \mathrm{d}\vartheta \mathrm{d}\varphi$. There is no need for special content, but the length of words should match the language. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. $\sin^2(\alpha) + \cos^2(\beta) = 1$. If you read this text, you will get no information $E = mc^2$. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. $\sqrt[n]{a} \cdot \sqrt[n]{b} = \sqrt[n]{ab}$. This text should contain all letters of the alphabet and it should be written in of the original language. $\frac{\sqrt[n]{a}}{\sqrt[n]{b}} = \sqrt[n]{\frac{a}{b}}$. There is no need for special content, but the length of words should match the language. $a\sqrt[n]{b} = \sqrt[n]{a^n b}$.

This is text bla bla. [2]

More testing on section 3.3.

Rango de activación: en el modelo original usamos ClippedReLU, asi que queremos que el rango vaya de 0..1 a 0..127.

Siendo $\boldsymbol{x}$, $\boldsymbol{w}$ y $\boldsymbol{b}$ los parámetros de una capa lineal sin cuantizar e $\boldsymbol{y}$ la salida de la misma, se tiene que:

$$\begin{aligned} \boldsymbol{y} &= \boldsymbol{x}\boldsymbol{w} + \boldsymbol{b} \\ s_a s_w \boldsymbol{y} &= (s_a \boldsymbol{x})(s_w \boldsymbol{w}) + s_a s_w \boldsymbol{b} \end{aligned} \tag{1}$$

$s_o((s_a \boldsymbol{x})(s_w \boldsymbol{w}) + s_a s_w \boldsymbol{b}) = s_a s_w s_o \boldsymbol{y}$

3.4 Network sparsity

o combinar con 3.2? poner graficos con la sparsity de cada feature set, decir que es muy esparso todo y que se podría mejorar aún más

[2] This is a example footnote

4 Training

4.1 Dataset

Lichess is a free online site to play chess, and it provides a database[1] with all the games played on the site. It consists of serveral compressed PGN files[2] splitted by month since 2013, that add up to 1.7 terabytes compressed. It contains over 5 billion games, that equates to around 200 billion positions. In reality, that many positions are too much to handle so I'll use only a fraction of them, but I restrict derived datasets to only take one sample per game, to maximize the diversity of positions.

decir que la data de entrenamiento no es la misma. SF original usaba datos de hand-crafted, hoy en dia usa de LC0. Nosotros usamos de Lichess, computada por el SF moderno

4.2 Methods

4.2.1 Stockfish evaluations

4.2.2 PQR triplets

This is an additional technique I wanted to try, described in [METER REF BLOG]. Remember that we are trying to obtain a function f (the model) to give an evaluation of a position. The method is based in the assumption that players make optimal or near-optimal moves most of the time, even if they are amateurs.

1. For two position in succession $p \to q$ observed in the game, we will have $f(p) \neq f(q)$.
2. Going from p, not to q, but to a *random* position $p \to r$, we must have $f(r) > f(q)$ because the random move is better for the next player and worse for the player that made the move.

... With infinite compute, f would be the result of running minimax to the end of the game, since minimax always finds optimal moves.

4.3 Metrics

asd [1]

[1] Lichess database: `https://database.lichess.org`

[2] Portable Game Notation: a textual format to store chess games (moves and metadata)

5 Engine implementation

5.1 Alpha-Beta

5.2 Prunes

6 Results

6.1 Active neurons

medir si hay feature sets que no usen neuronas, que esto disparo el uso de HalfTopK

7 Final words

7.1 Conclusions

7.2 Future work

References

[1] Yu Nasu. “NNUE: Efficiently Updatable Neural-Network-based Evaluation Functions for Computer Shogi”. In: *Ziosoft Computer Shogi Club* (2018). URL: https://www.apply.computer-shogi.org/wcsc28/appeal/the_end_of_genesis_T.N.K.evolution_turbo_type_D/nnue.pdf.