

Feature set analysis for chess NN networks

Tesis de Licenciatura

Martín Emiliano Lombardo

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

2024



Introducción
●○○○○○○○○○

Engine
○

Feature set
○○○○○○○○○○

ЭУММ (NNUE)
○○○○○

Training
○

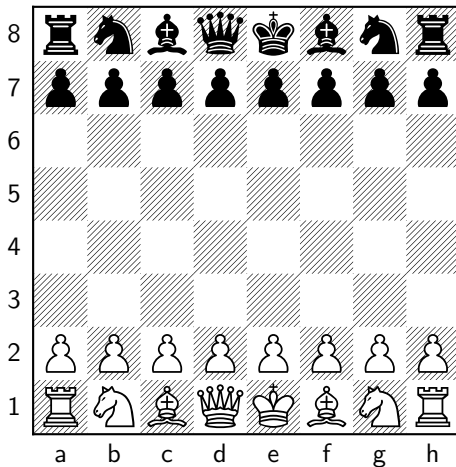
Experimentos
○○○○○○○

Conclusión
○○

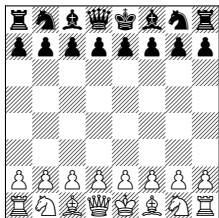
Introducción

Ajedrez

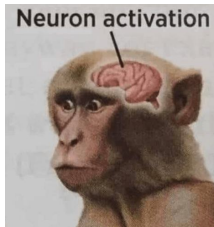
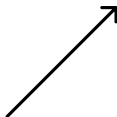
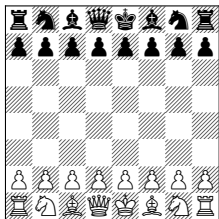
- Dos jugadores
- Suma cero



Humano vs. Computadora

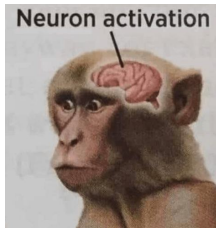
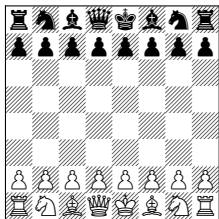


Humano vs. Computadora



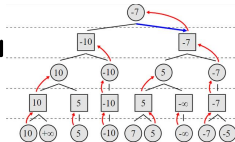
→ e2e4

Humano vs. Computadora



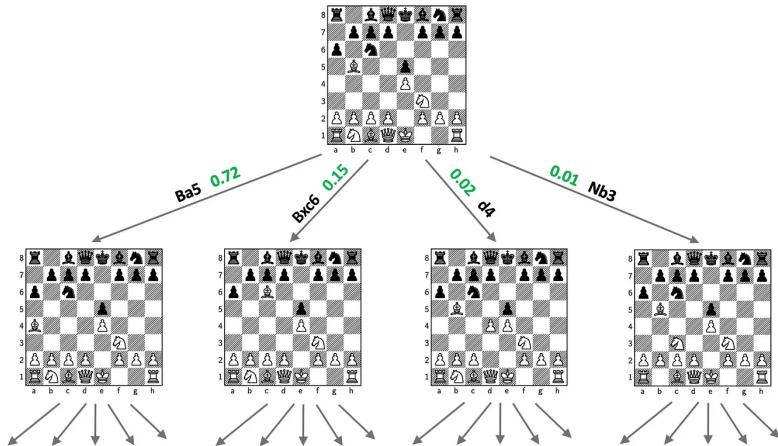
→ e2e4

Chess Engine



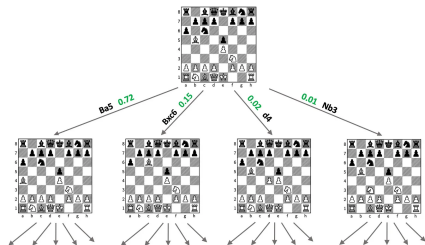
→ e2e4

Ajedrez como árbol



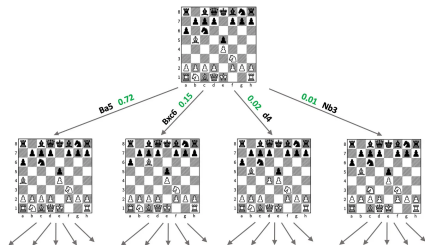
Motores de ajedrez (Chess Engines)

- Exploran el árbol de juego (Minimax, MCTS, etc.)



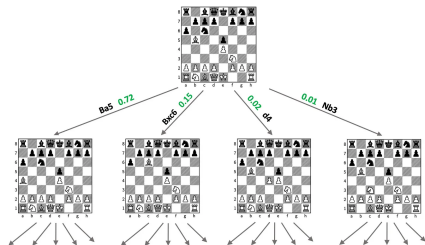
Motores de ajedrez (Chess Engines)

- Exploran el árbol de juego (Minimax, MCTS, etc.)
- Utilizan funciones de evaluación en las hojas




Motores de ajedrez (Chess Engines)

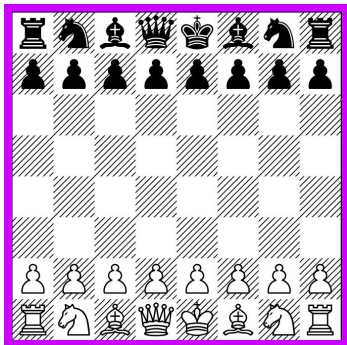
- Exploran el árbol de juego (Minimax, MCTS, etc.)
- Utilizan funciones de evaluación en las hojas
- La evaluación se propaga hacia arriba, según el algoritmo



Función de evaluación

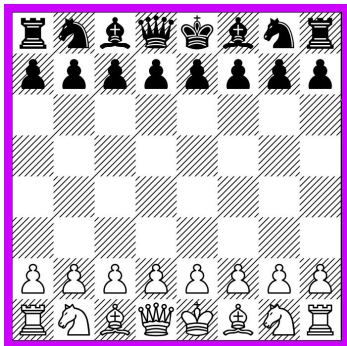
$$f \left(\begin{array}{c} \text{Chessboard state} \end{array} \right) = 5$$


(adelanto) Feature set: ¿Cómo transformar la posición a un vector?



$$f(?) = ?$$

(adelanto) Feature set: ¿Cómo transformar la posición a un vector?



feature set!

$$f(?) = ?$$

Motores de ajedrez (breve historia)

Plan

asdasd

- Text visible on slide 1

asdasd

Plan

asdasd

- Text visible on slide 1
- Text visible on slide 2

asdasd

Plan

asdasd

- Text visible on slide 1
- Text visible on slide 2
- Text visible on slide 3

asdasd

Plan

asdasd

- Text visible on slide 1
- Text visible on slide 2
- Text visible on slide 4

asdasd

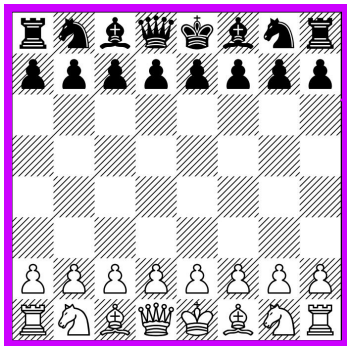
Contenido

- 1 Introducción
- 2 Engine
- 3 Feature set
 - Motivación
 - Definición
 - Operadores
 - Feature sets conocidos
 - Resumen
- 4 $\exists U \forall V$ (NNUE)
- 5 Training
- 6 Experimentos
- 7 Conclusión

Engine

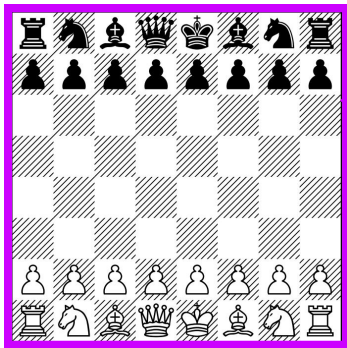
Feature set

¿Cómo transformar la posición a un vector?



$$f(?) = ?$$

¿Cómo transformar la posición a un vector?



feature set!

$$f(?) = ?$$

Definición

Un **feature set** S_P se define con un conjunto S y un predicado asociado $P(e)$, donde:

- S es un conjunto de conceptos (rol, color, celda, número, etc.).
- $P(e)$ es un predicado que determina si e está presente (o *activo*) en la posición (implícita).

Definición

Un **feature set** S_P se define con un conjunto S y un predicado asociado $P(e)$, donde:

- S es un conjunto de conceptos (rol, color, celda, número, etc.).
- $P(e)$ es un predicado que determina si e está presente (o *activo*) en la posición (implícita).
- Cada elemento en S_P es un *feature*.

Definición

Un **feature set** S_P se define con un conjunto S y un predicado asociado $P(e)$, donde:

- S es un conjunto de conceptos (rol, color, celda, número, etc.).
- $P(e)$ es un predicado que determina si e está presente (o *activo*) en la posición (implícita).
- Cada elemento en S_P es un *feature*.
- Cada *feature* es un valor en el vector de entrada, valiendo 1 si está *activo* y 0 si no.

Ejemplos de S

Información posicional:

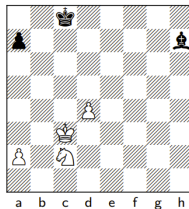
$$\text{FILES} = \{a, b, \dots, h\}$$
$$\text{RANKS} = \{1, 2, \dots, 8\}$$
$$\text{SQUARES} = \{a1, a2, \dots, h8\}$$

8	a8	b8	c8	d8	e8	f8	g8	h8
7	a7	b7	c7	d7	e7	f7	g7	h7
6	a6	b6	c6	d6	e6	f6	g6	h6
5	a5	b5	c5	d5	e5	f5	g5	h5
4	a4	b4	c4	d4	e4	f4	g4	h4
3	a3	b3	c3	d3	e3	f3	g3	h3
2	a2	b2	c2	d2	e2	f2	g2	h2
1	a1	b1	c1	d1	e1	f1	g1	h1
	a	b	c	d	e	f	g	h

Información sobre las piezas:

$$\text{ROLES} = \{ \text{♙ Pawn}, \text{♘ Knight}, \text{♚ Bishop}, \text{♖ Rook}, \text{♕ Queen}, \text{♔ King} \}$$
$$\text{COLORS} = \{ \text{○ White}, \text{● Black} \}$$

Ejemplo completo



	Feature set	
	$(\text{FILES} \times \text{COLORS})_P$	$(\text{FILES} \times \text{ROLES})_Q$
Active features	$\langle a, \bigcirc \rangle, \langle a, \bullet \rangle, \langle c, \bullet \rangle,$ $\langle c, \bigcirc \rangle, \langle d, \bigcirc \rangle, \langle h, \bullet \rangle$	$\langle a, \text{♙} \rangle, \langle c, \text{♔} \rangle, \langle c, \text{♚} \rangle,$ $\langle d, \text{♙} \rangle, \langle h, \text{♗} \rangle$

$P(\langle f, c \rangle)$: there is a piece in file f with color c .

$Q(\langle f, r \rangle)$: there is a piece in file f with role r .

Operación: Suma \oplus (concatenación)

Hay veces que es útil combinar información de dos *feature sets*

Operación: Suma \oplus (concatenación)

Hay veces que es útil combinar información de dos *feature sets*

S_P, T_Q : feature sets

$$S_P \oplus T_Q = (S \cup T)_R$$

$$\text{where } R(e) = \begin{cases} P(e) & \text{if } e \in S \\ Q(e) & \text{if } e \in T \end{cases}$$

Operación: Producto \times (and)

$$S_P \times T_Q = (S \times T)_R$$

where $R(\langle e_0, e_1 \rangle) = P(e_0) \wedge Q(e_1)$

Feature set: ALL

La codificación más natural de una posición de ajedrez

$$\text{ALL} : (\text{SQUARES} \times \text{ROLES} \times \text{COLORS})_P$$

$P(\langle s, r, c \rangle)$: there is a piece in square s with role r and color c

Feature set: ALL

La codificación más natural de una posición de ajedrez

$ALL : (SQUARES \times ROLES \times COLORS)_P$
 $P(\langle s, r, c \rangle)$: there is a piece in square s with role r and color c

- Es pequeño: $64 \times 6 \times 2 = 768$ *features*

Feature set: ALL

La codificación más natural de una posición de ajedrez

$$ALL : (SQUARES \times ROLES \times COLORS)_P$$

$P(\langle s, r, c \rangle)$: there is a piece in square s with role r and color c

- Es pequeño: $64 \times 6 \times 2 = 768$ *features*
- Es completo: contiene toda la información de la posición

Feature set: ALL

La codificación más natural de una posición de ajedrez

$$ALL : (SQUARES \times ROLES \times COLORS)_P$$

$P(\langle s, r, c \rangle)$: there is a piece in square s with role r and color c

- Es pequeño: $64 \times 6 \times 2 = 768$ *features*
- Es completo: contiene toda la información de la posición
- Es muy rápido computar cuáles *features* están activas

Feature set: KING-ALL ó “KA”

Los engines modernos usan variaciones del siguiente feature set.
Permite entender la posición en relación a la posición del rey:

$$\text{KING-ALL} = \text{SQUARE}_K \times \text{ALL}$$

$K(s)$: s is the square of the king of the side to move

Feature set: KING-ALL ó “KA”

Los engines modernos usan variaciones del siguiente feature set.
Permite entender la posición en relación a la posición del rey:

$$\text{KING-ALL} = \text{SQUARE}_K \times \text{ALL}$$

$K(s)$: s is the square of the king of the side to move

- Es grande: $64 \times 768 = 49152$ *features*

Feature set: KING-ALL ó “KA”

Los engines modernos usan variaciones del siguiente feature set.
Permite entender la posición en relación a la posición del rey:

$$\text{KING-ALL} = \text{SQUARE}_K \times \text{ALL}$$

$K(s)$: s is the square of the king of the side to move

- Es grande: $64 \times 768 = 49152$ *features*
- Es muy rápido como ALL

Feature set: KING-ALL ó “KA”

Los engines modernos usan variaciones del siguiente feature set.
Permite entender la posición en relación a la posición del rey:

$$\text{KING-ALL} = \text{SQUARE}_K \times \text{ALL}$$

$K(s)$: s is the square of the king of the side to move

- Es grande: $64 \times 768 = 49152$ *features*
- Es muy rápido como ALL
- Entrenarlo require un dataset más grande y lleva más tiempo (no me meto acá)

Feature sets: resumen

- **S** : set of concepts (roles, colors, squares, files, ranks, etc.).
- **$P(e)$** : predicate that defines when the feature e is present in the (implicit) position.
- **S_P** : a feature set. Every element in S_P is a feature. Features that satisfy P are *active*.
- $S_P \times T_Q = (S \times T)_R$ where $R(\langle e_0, e_1 \rangle) = P(e_0) \wedge Q(e_1)$
- $S_P \oplus T_Q = (S \cup T)_R$ where $R(e) = \begin{cases} P(e) & \text{if } e \in S \\ Q(e) & \text{if } e \in T \end{cases}$

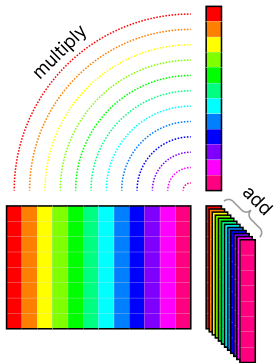
ENNE (NNUE)

ΕUUN: Efficiently Updatable Neural Networks

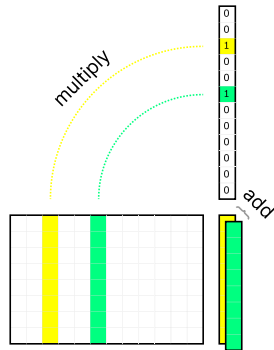
EUNN: Neural Networks

- El input es un vector one-hot generado por el *feature set*.
 - Debe tener pocos *features* activos (rara): introduce una cota superior.
- La red es una *feedforward* clásica con dos capas ocultas.

Linear layer



(g) Linear layer



(h) Linear layer with sparse inputs

Figure: Linear layer operation comparison. Figures from [18].

EUNE: Efficient Updates

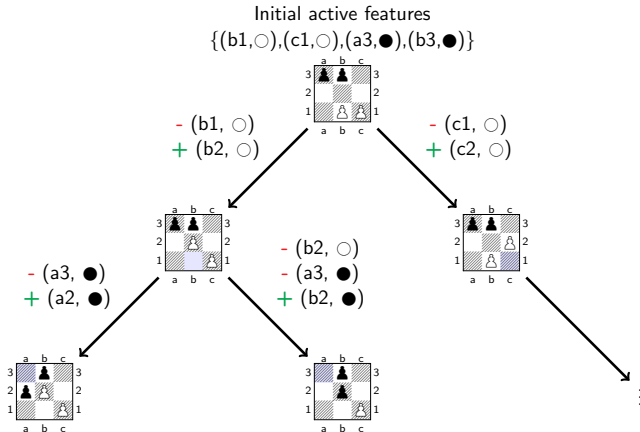


Figure: Partial tree of feature updates (removals and additions) for $(\text{SQUARES} \times \text{COLORS})$ (white's point of view) in a simplified 3x3 pawn-only board.

ΕUΜM: Tradeoff

motivacion comparacion de burns

Training

Experimentos

Setup de training

¿Qué hay que definir para entrenar una red?

Setup de training

¿Qué hay que definir para entrenar una red?

- **Feature set:** determina la codificación y los patrones que se pueden aprender

Setup de training

¿Qué hay que definir para entrenar una red?

- **Feature set:** determina la codificación y los patrones que se pueden aprender
- **Dataset:** datos de entrenamiento, visto anteriormente

Setup de training

¿Qué hay que definir para entrenar una red?

- **Feature set:** determina la codificación y los patrones que se pueden aprender
- **Dataset:** datos de entrenamiento, visto anteriormente
- **Arquitectura de la red:** el tamaño de cada capa; L_1 y L_2

Setup de training

¿Qué hay que definir para entrenar una red?

- **Feature set:** determina la codificación y los patrones que se pueden aprender
- **Dataset:** datos de entrenamiento, visto anteriormente
- **Arquitectura de la red:** el tamaño de cada capa; L_1 y L_2
- **Método de entrenamiento:** PQR/target scores; determina el formato de las muestras y la loss function

Setup de training

¿Qué hay que definir para entrenar una red?

- **Feature set:** determina la codificación y los patrones que se pueden aprender
- **Dataset:** datos de entrenamiento, visto anteriormente
- **Arquitectura de la red:** el tamaño de cada capa; L_1 y L_2
- **Método de entrenamiento:** PQR/target scores; determina el formato de las muestras y la loss function
- **Hiperparámetros:** learning rate, batch size, epochs, etc.

Setup de evaluación

¿Cómo evalúo el performance de una red entrenada?

Setup de evaluación

¿Cómo evalúo el performance de una red entrenada?

- **Loss** (train y val.): indica la calidad de las predicciones.
 - Permite detectar overfitting y otros problemas

Setup de evaluación

¿Cómo evalúo el performance de una red entrenada?

- **Loss** (train y val.): indica la calidad de las predicciones.
 - Permite detectar overfitting y otros problemas
- **Puzzle accuracy**: porcentaje de movimientos acertados en puzzles de Lichess.
 - Sólo hay un movimiento correcto
 - Proxy (muy malo) de la fuerza de la red

Setup de evaluación

¿Cómo evalúo el performance de una red entrenada?

- **Loss** (train y val.): indica la calidad de las predicciones.
 - Permite detectar overfitting y otros problemas
- **Puzzle accuracy**: porcentaje de movimientos acertados en puzzles de Lichess.
 - Sólo hay un movimiento correcto
 - Proxy (muy malo) de la fuerza de la red
- **Elo relativo**: la medida más común para comparar engines.
 - Se realizan torneos de 100ms por movimiento
 - El elo es calculado a partir de Ordo

Baseline: motivación

Busco fijar el setup de entrenamiento con valores razonables

Baseline: motivación

Busco fijar el setup de entrenamiento con valores razonables

- El feature set va a cambiar cada experimento

Baseline: motivación

Busco fijar el setup de entrenamiento con valores razonables

- El feature set va a cambiar cada experimento
- El dataset está fijo

Baseline: motivación

Busco fijar el setup de entrenamiento con valores razonables

- El feature set va a cambiar cada experimento
- El dataset está fijo
- El método de entrenamiento principal es *target scores*

Baseline: motivación

Busco fijar el setup de entrenamiento con valores razonables

- El feature set va a cambiar cada experimento
- El dataset está fijo
- El método de entrenamiento principal es *target scores*

Entonces queda por determinar...

- La arquitectura de la red (L_1 y L_2)

Baseline: motivación

Busco fijar el setup de entrenamiento con valores razonables

- El feature set va a cambiar cada experimento
- El dataset está fijo
- El método de entrenamiento principal es *target scores*

Entonces queda por determinar...

- La arquitectura de la red (L_1 y L_2)
- Los hiperparámetros

Baseline: hiperparámetros

Los hiperparámetros fueron seleccionados en base al trainer oficial de Stockfish:

Baseline: hiperparámetros

Los hiperparámetros fueron seleccionados en base al trainer oficial de Stockfish:

- **Learning rate:** 0.0005

Baseline: hiperparámetros

Los hiperparámetros fueron seleccionados en base al trainer oficial de Stockfish:

- **Learning rate:** 0.0005
- **Exponential decay:** 0.99

Baseline: hiperparámetros

Los hiperparámetros fueron seleccionados en base al trainer oficial de Stockfish:

- **Learning rate:** 0.0005
- **Exponential decay:** 0.99
- **Batch size:** 16384

Baseline: hiperparámetros

Los hiperparámetros fueron seleccionados en base al trainer oficial de Stockfish:

- **Learning rate:** 0.0005
- **Exponential decay:** 0.99
- **Batch size:** 16384
- **Epoch size:** 100 million

Baseline: hiperparámetros

Los hiperparámetros fueron seleccionados en base al trainer oficial de Stockfish:

- **Learning rate:** 0.0005
- **Exponential decay:** 0.99
- **Batch size:** 16384
- **Epoch size:** 100 million
 - cada epoch realiza 6104 batches

Baseline: hiperparámetros

Los hiperparámetros fueron seleccionados en base al trainer oficial de Stockfish:

- **Learning rate:** 0.0005
- **Exponential decay:** 0.99
- **Batch size:** 16384
- **Epoch size:** 100 million
 - cada epoch realiza 6104 batches
- **Epochs:** 256
 - cada run observa *25.6 billion* samples

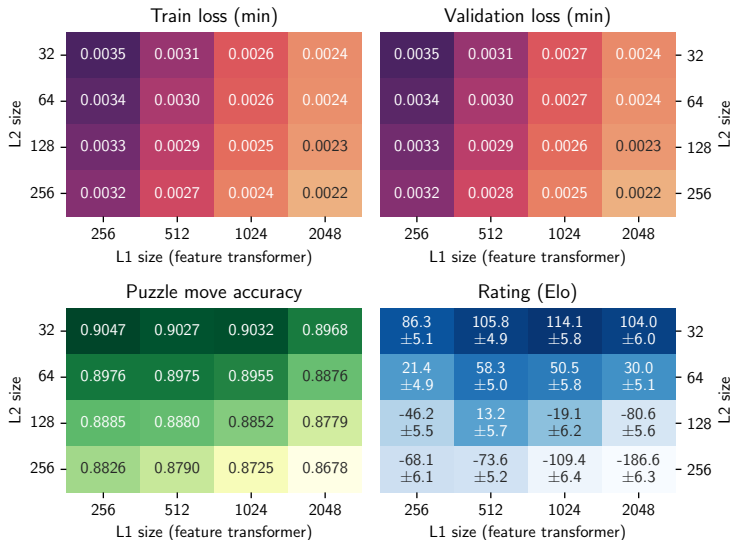
Baseline: experimento

Sólo queda buscar parámetros L_1 y L_2 razonables. Realizo una búsqueda en grilla con:

- $L_1 \in \{256, 512, 1024, 2048\}$
- $L_2 \in \{32, 64, 128, 256\}$

El feature set a utilizar es $ALL[768]$.

Baseline: resultados



Conclusión

Ajedrez

■ asdasd