

Feature set analysis for chess ΣNN networks

Tesis de Licenciatura

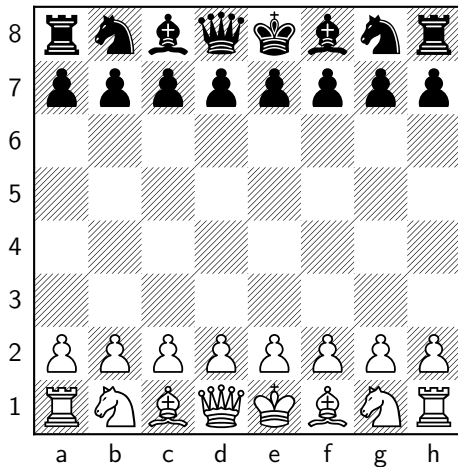
Martín Emiliano Lombardo

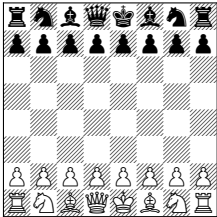
Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

2024

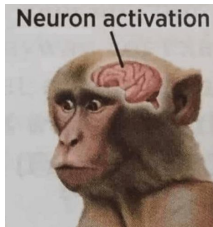
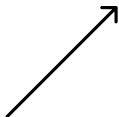
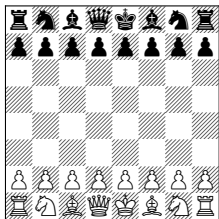


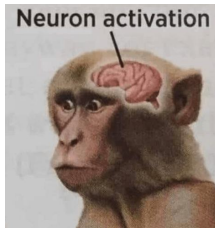
Introducción





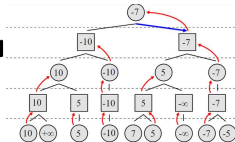
Humano vs. Computadora

 $\longrightarrow e2e4$



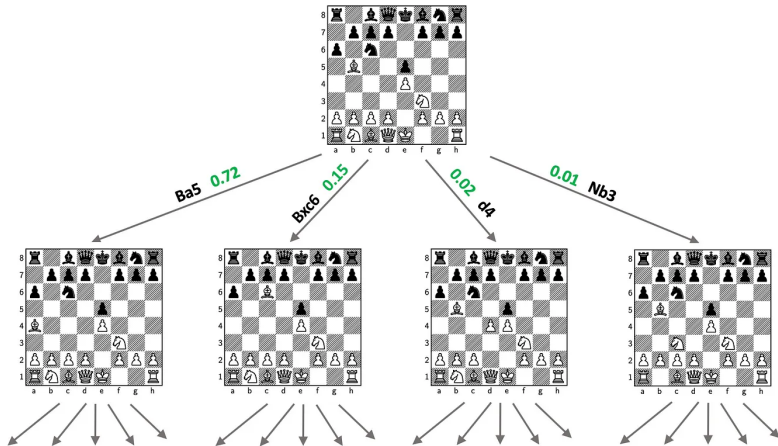
e2e4

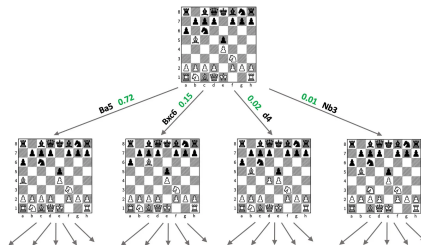
Chess Engine



e2e4

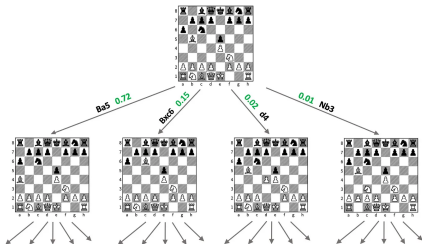
Ajedrez como árbol





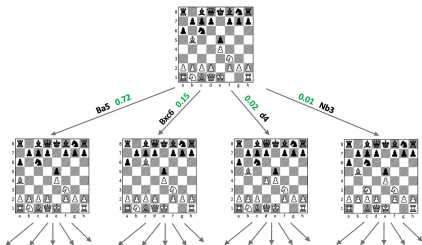
Motores de ajedrez (Chess Engines)

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

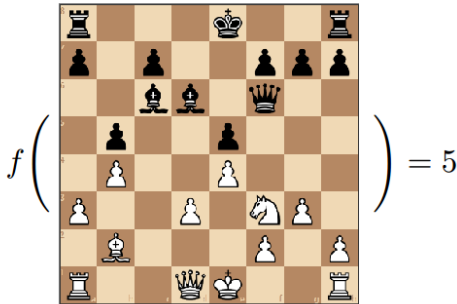


Motores de ajedrez (Chess Engines)

- 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 o “eval”



Intentan resumir todo el subárbol en un solo número.
En general son creadas *artesanalmente*

Motores de ajedrez (breve historia)

- **1950s:** Se desarrollan los primeros *algoritmos* de ajedrez

- **1950s:** Se desarrollan los primeros *algoritmos* de ajedrez
- **1960s+:** Aparecen los primeros *motores de ajedrez*, lentos y débiles

Motores de ajedrez (breve historia)

- **1950s**: Se desarrollan los primeros *algoritmos* de ajedrez
- **1960s+**: Aparecen los primeros *motores de ajedrez*, lentos y débiles
- **1997** (hito): IBM DeepMind vence a Garry Kasparov en un torneo

Motores de ajedrez (breve historia)

- **1950s:** Se desarrollan los primeros *algoritmos* de ajedrez
- **1960s+:** Aparecen los primeros *motores de ajedrez*, lentos y débiles
- **1997** (hito): IBM DeepMind vence a Garry Kasparov en un torneo
- **2017 y 2018:** Google DeepMind publica AlphaGo Zero y su sucesor AlphaZero
 - se reemplaza la función de evaluación por una red neuronal

Motores de ajedrez (breve historia)

- **1950s:** Se desarrollan los primeros *algoritmos* de ajedrez
- **1960s+:** Aparecen los primeros *motores de ajedrez*, lentos y débiles
- **1997** (hito): IBM DeepMind vence a Garry Kasparov en un torneo
- **2017 y 2018:** Google DeepMind publica AlphaGo Zero y su sucesor AlphaZero
 - se reemplaza la función de evaluación por una red neuronal
- **2018:** Yu Nasu introduce las redes EUNN para Shogi

Plan de la tesis

El objetivo principal es **proponer y evaluar novedosos feature sets**.

Plan de la tesis

El objetivo principal es **proponer y evaluar novedosos feature sets**. Además, **probar una técnica de entrenamiento** no convencional.

Plan de la tesis

El objetivo principal es **proponer y evaluar novedosos feature sets**. Además, **probar una técnica de entrenamiento** no convencional.

El plan de la presentación es el siguiente:

- Implementación de un motor

Plan de la tesis

El objetivo principal es **proponer y evaluar novedosos feature sets**. Además, **probar una técnica de entrenamiento** no convencional.

El plan de la presentación es el siguiente:

- Implementación de un motor
- Feature sets

Plan de la tesis

El objetivo principal es **proponer y evaluar novedosos feature sets**. Además, **probar una técnica de entrenamiento** no convencional.

El plan de la presentación es el siguiente:

- Implementación de un motor
- Feature sets
- NNUE

Plan de la tesis

El objetivo principal es **proponer y evaluar novedosos feature sets**. Además, **probar una técnica de entrenamiento** no convencional.

El plan de la presentación es el siguiente:

- Implementación de un motor
- Feature sets
- NNUE
- Training

Plan de la tesis

El objetivo principal es **proponer y evaluar novedosos feature sets**. Además, **probar una técnica de entrenamiento** no convencional.

El plan de la presentación es el siguiente:

- Implementación de un motor
- Feature sets
- NNUE
- Training
- Experimentos

Engine

Construir un motor clásico de ajedrez, con optimizaciones clásicas, pero que use NNUE para evaluar posiciones.

Engine

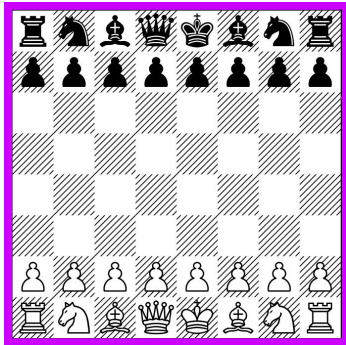
Primera idea: evaluar todas las posiciones a las que me puedo mover y elegir la mejor. Extender la idea recursivamente. Ojo que ahora está min... min nodes, max nodes

Minimax

el dibujo

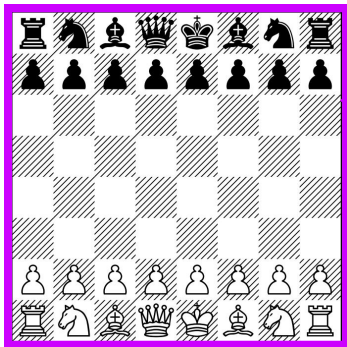
Feature set

¿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:

$$\text{FILES} = \{a, b, \dots, h\}$$

$$\text{RANKS} = \{1, 2, \dots, 8\}$$

$$\text{SQUARES} = \{a1, a2, \dots, h8\}$$

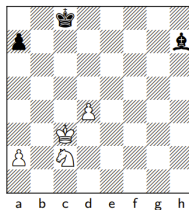
| | | | | | | | | |
|---|----|----|----|----|----|----|----|----|
| 8 | a8 | b8 | c8 | d8 | e8 | f8 | g8 | h8 |
| 7 | a7 | b7 | c7 | d7 | e7 | f7 | g7 | h7 |
| 6 | a6 | b6 | c6 | d6 | e6 | f6 | g6 | h6 |
| 5 | a5 | b5 | c5 | d5 | e5 | f5 | g5 | h5 |
| 4 | a4 | b4 | c4 | d4 | e4 | f4 | g4 | h4 |
| 3 | 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)

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

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

Feature set: ALL

La codificación más natural de una posición de ajedrez

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

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

Feature set: ALL

La codificación más natural de una posición de ajedrez

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

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

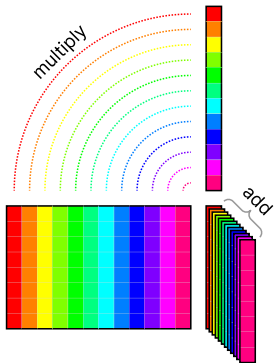
ENNE (NNUE)

EUNN: Efficiently Updatable Neural Networks

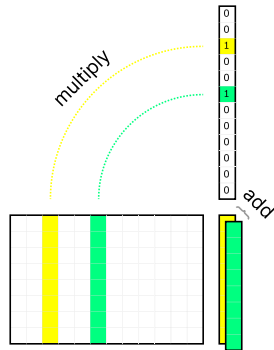
EUNN: Neural Networks

- 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

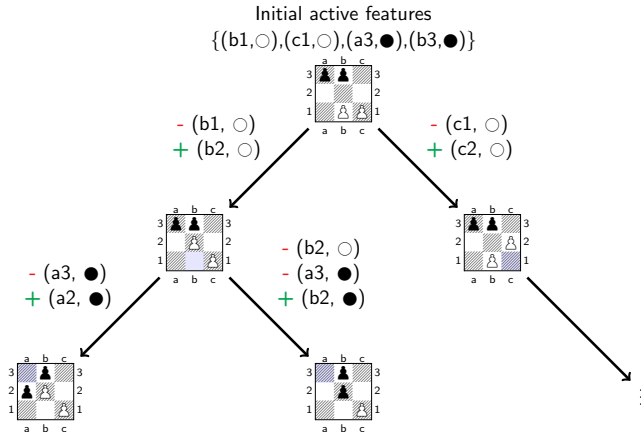


Figure: Partial tree of feature updates (removals and additions) for $(\text{SQUARES} \times \text{COLORS})$ (white's point of view) in a simplified 3x3 pawn-only board.

EUNN: Tradeoff

motivacion comparacion de burns

Training

Un blogpost de 2014 por Erik Bernhardsson propone entrenar una red utilizando dos principios:

Experimentos

Setup de training

Recapitulando... ¿Qué hay que definir para entrenar una red?

Setup de training

Recapitulando... ¿Qué hay que definir para entrenar una red?

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

Setup de training

Recapitulando... ¿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

Recapitulando... ¿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

Recapitulando... ¿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

Recapitulando... ¿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
- **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

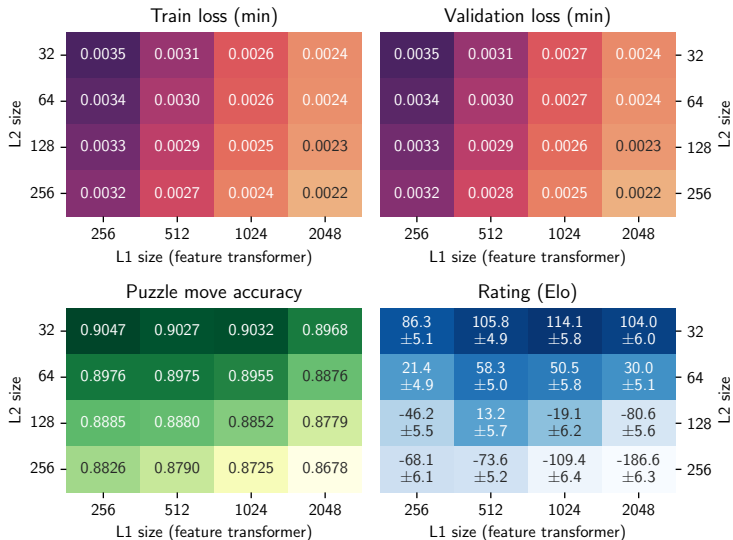
Baseline: experimento

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



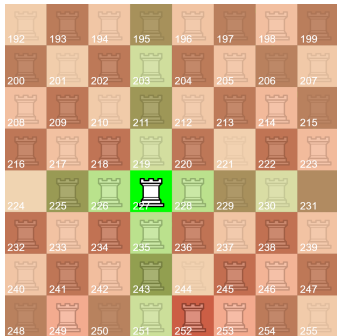
Baseline: conclusión

- **L2=32.** El performance cae dramáticamente si L2 aumenta, utilizo el más bajo.
 - Sería buena idea probar valores más chicos de L2.

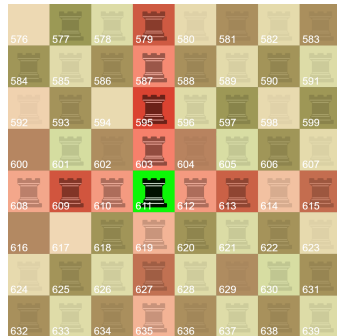
Baseline: conclusión

- **L2=32.** El performance cae dramáticamente si L2 aumenta, utilizo el más bajo.
 - Sería buena idea probar valores más chicos de L2.
- **L1=512.** Es el mejor valor para L2=64 y L2=128, y en margen de error para L2=32.
 - Además es el más rápido de entrenar.

Axis encoding: motivación



(a) ○ White



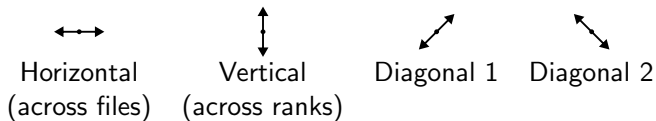
(b) ● Black

Figure: Weights of **a neuron** in the L1 layer, which are connected to features in ALL where the role is ♖ Rook. The intensity represents the weight value, and the color represents the sign (although not relevant).

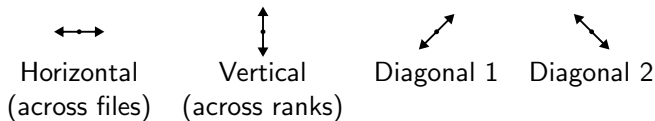
Axis encoding: motivación





La red detecta patrones parecidos a los movimientos de las piezas.

Axis encoding: experimento



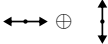


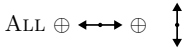


Axis encoding: experimento



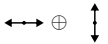

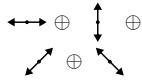
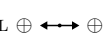
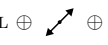
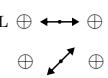
| Depiction | Block name | Definition | Number of features |
|---|------------|--|--------------------|
|  | H | $(\text{FILES} \times \text{ROLES} \times \text{COLORS})_P$ | 96 |
|  | V | $(\text{RANKS} \times \text{ROLES} \times \text{COLORS})_P$ | 96 |
|  | D1 | $(\text{DIAGS1} \times \text{ROLES} \times \text{COLORS})_P$ | 180 |
|  | D2 | $(\text{DIAGS2} \times \text{ROLES} \times \text{COLORS})_P$ | 180 |

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

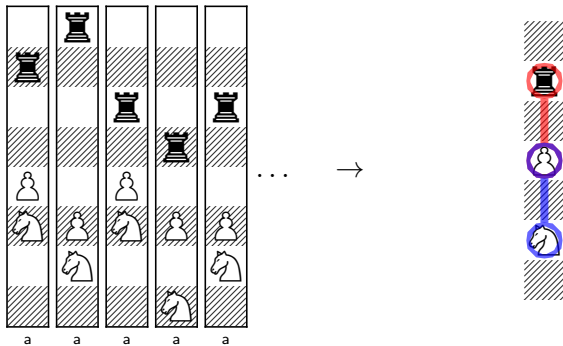
Axis encoding: experimento

| Depiction | Feature set | Number of features |
|---|---|--------------------|
|  | $H \oplus V$ | 192 |
|  | $D1 \oplus D2$ | 360 |
|  | $H \oplus V \oplus D1 \oplus D2$ | 552 |
|  | $ALL \oplus H \oplus V$ | 960 |
|  | $ALL \oplus D1 \oplus D2$ | 1128 |
|  | $ALL \oplus H \oplus V \oplus D1 \oplus D2$ | 1320 |

Axis encoding: resultados

| Feature set | Number of features | Val. loss <i>min</i> | Rating <i>elo (rel. to ALL)</i> | Puzzles <i>move acc.</i> |
|--|--------------------|-------------------------|------------------------------------|-----------------------------|
|  | 192 | 0.005810 | -384.3 ± 5.1 | 0.8618 |
|  | 360 | 0.006707 | -444.1 ± 5.1 | 0.8517 |
|  | 552 | 0.003907 | -183.5 ± 4.1 | 0.8748 |
| ALL (reference) | 768 | 0.003134 | 0.0 | 0.8865 |
| ALL \oplus  | 960 | 0.003082 | -27.1 ± 4.1 | 0.8851 |
| ALL \oplus  | 1128 | 0.003087 | -26.1 ± 3.8 | 0.8814 |
| ALL \oplus  | 1320 | 0.003067 | -58.7 ± 3.7 | 0.8766 |

Pairwise axes: motivación



Configuraciones distintas,
situaciones similares

Las mismas dos features
(par rojo y par azul)

Pairwise axes: motivación



Comparando con el experimento anterior, es más específico en vez de más general:

“there is a ○ White ♖ Rook in the 4th rank”

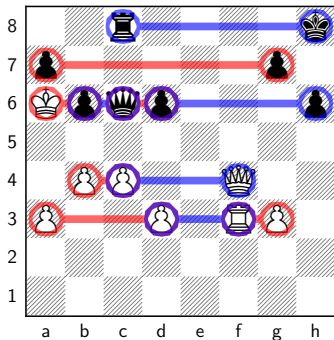
vs.

“there is a ● Black ♜ Rook next to a ○ White ♙ Pawn in the ‘a’ file”

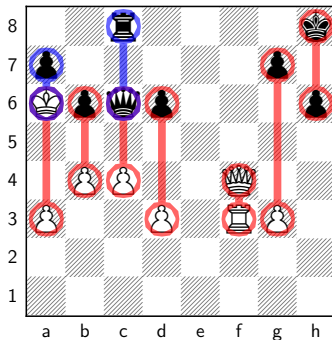
Pairwise axes: experimento

| D. | Block name | Definition | Num. of features |
|---|------------|--|------------------|
|  | PH | $(\text{RANKS} \times (\text{ROLES} \times \text{COLORS}) \times (\text{ROLES} \times \text{COLORS}))_P$ $P(\langle r, r_1, c_1, r_2, c_2 \rangle)$: there is a piece in rank r with role r_1 and color c_1 to the left of a piece with role r_2 and color c_2 | 1152 |
|  | PV | $(\text{FILES} \times (\text{ROLES} \times \text{COLORS}) \times (\text{ROLES} \times \text{COLORS}))_Q$ $Q(\langle f, r_1, c_1, r_2, c_2 \rangle)$: there is a piece in file f with role r_1 and color c_1 below a piece with role r_2 and color c_2 | 1152 |

Pairwise axes: experimento



Pairwise horizontal (PH)







Pairwise vertical (PV)


Pairwise axes: experimento

Los feature sets a entrenar son:

- $ALL \oplus PH$ (1920 features)
- $ALL \oplus PV$ (1920 features)
- $ALL \oplus PH \oplus PV$ (3072 features)

Pairwise axes: resultados

| Feature set | Number of features | Val. loss <i>min</i> | Rating <i>elo (rel. to ALL)</i> |
|---|-----------------------|-------------------------|------------------------------------|
| ALL (reference) | 768 | 0.003134 | 0.0 |
| ALL \oplus  | 1920 | 0.003033 | -38.2 \pm 4.8 |
| ALL \oplus  | 1920 | 0.002946 | -8.4 \pm 5.0 |
| ALL \oplus  \oplus  | 3072 | 0.002868 | -37.6 \pm 4.9 |

- Reducir el número de pairs puede llevar a una mejora por sobre ALL (ej. )

Mobility: motivación

- La *movilidad* en ajedrez es una medida de la cantidad de movimientos que puede hacer un jugador en una posición.

Mobility: motivación

- La *movilidad* en ajedrez es una medida de la cantidad de movimientos que puede hacer un jugador en una posición.
- Un paper de Eliot Slater (1950) mostró que hay una correlación entre la movilidad de un jugador y la cantidad de partidas ganadas.

Mobility: motivación

- La *movilidad* en ajedrez es una medida de la cantidad de movimientos que puede hacer un jugador en una posición.
- Un paper de Eliot Slater (1950) mostró que hay una correlación entre la movilidad de un jugador y la cantidad de partidas ganadas.
- Se usa en funciones de evaluación hechas a mano.

Mobility: motivación

- La *movilidad* en ajedrez es una medida de la cantidad de movimientos que puede hacer un jugador en una posición.
- Un paper de Eliot Slater (1950) mostró que hay una correlación entre la movilidad de un jugador y la cantidad de partidas ganadas.
- Se usa en funciones de evaluación hechas a mano.
- Propongo agregar movilidad como features en la red.

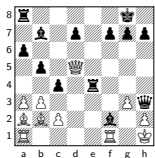
Mobility: experimento

Hay dos maneras de codificar la movilidad:

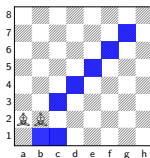
- Bitsets (por rol/color)
- Cantidades (por rol/color)

Mobility: experimento (bitsets)

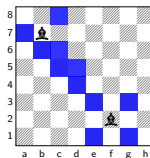
Los features proveen **las celdas** a las que una pieza de determinado rol/color puede moverse.



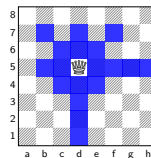
Board



○ White
♗ Bishop



● Black
♘ Bishop









○ White
♕ Queen

...

La cantidad de features es $64 \times 6 \times 2 = 768$, la misma que ALL.

Mobility: experimento (counts)

Los features proveen **la cantidad de celdas** a las que una pieza de determinado rol/color puede moverse. Esto reduce la cantidad de features significativamente.

| Piece role | Min | Max |
|--|-----|-----|
|  Pawn | 0 | 8+ |
|  Knight | 0 | 15+ |
|  Bishop | 0 | 16+ |
|  Rook | 0 | 25+ |
|  Queen | 0 | 25+ |
|  King | 0 | 8 |

Mobility: experimento (counts)

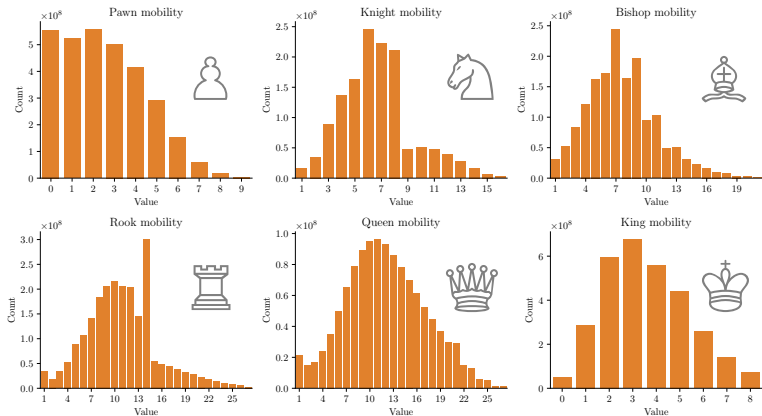


Figure: Total mobility values for each piece on the board. Computed using 2 billion boards. The value 0 for the ♞ Knight, ♗ Bishop, ♖ Rook, and ♔ Queen has been excluded from the plot, as it is very common.

Mobility: experimento

| Block name | Definition | Number of features |
|------------|---|--------------------|
| MB | $(\text{SQUARES} \times \text{ROLES} \times \text{COLORS})_P$ $P(\langle s, r, c \rangle)$: there is a piece of role r and color c that can move to square s | 768 |
| MC | $(\{0, 1, \dots\} \times \text{ROLES} \times \text{COLORS})_P$ $P(\langle m, r, c \rangle)$: the value of mobility for a piece of role r and color c is m | 206 |

Los feature sets a entrenar son: $\text{ALL} \oplus \text{MB}$ (1536 features) y $\text{ALL} \oplus \text{MC}$ (974 features).

Mobility: resultados

Table: Mobility encodings results

| Feature set | Number of features | Val. loss <i>min</i> | Rating <i>elo (rel. to ALL)</i> |
|-----------------|--------------------|-------------------------|------------------------------------|
| ALL (reference) | 768 | 0.003134 | 0.0 |
| ALL \oplus MB | 1536 | 0.002824 | -260.9 \pm 5.4 |
| ALL \oplus MC | 974 | 0.003032 | -280.9 \pm 5.6 |

- Las predicciones mejoran muy poco (el loss no se reduce tanto).
- Por ende, el costo de las actualizar los features es más alto al beneficio que aportan.








Mobility: resultados

Table: Mobility encodings results

| Feature set | Number of features | Val. loss <i>min</i> | Rating <i>elo (rel. to ALL)</i> |
|-----------------|--------------------|-------------------------|------------------------------------|
| ALL (reference) | 768 | 0.003134 | 0.0 |
| ALL \oplus MB | 1536 | 0.002824 | -260.9 \pm 5.4 |
| ALL \oplus MC | 974 | 0.003032 | -280.9 \pm 5.6 |

- Las predicciones mejoran muy poco (el loss no se reduce tanto).
- Por ende, el costo de las actualizar los features es más alto al beneficio que aportan.
- MB tiene más updates que MC, pero menor loss que compensa.

Feature set statistics

| Depiction | Feature block | Number of features | Average features... | | |
|---|---------------|--------------------|------------------------|-------------------|---------------------|
| | | | active per position | added per move | removed per move |
|  | ALL | 768 | 14.68 | 0.98 | 0.60 |
|  | H | 96 | 14.68 | 0.60 | 0.43 |
|  | V | 96 | 14.68 | 0.61 | 0.43 |
|  | D1 | 180 | 14.68 | 0.77 | 0.52 |
|  | D2 | 180 | 14.68 | 0.77 | 0.52 |
|  | PH | 1152 | 8.23 | 0.92 | 0.57 |
|  | PV | 1152 | 8.30 | 0.83 | 0.53 |
| MB | MB | 768 | 48.93 | 5.68 | 4.35 |
| MC | MC | 206 | 12.00 | 2.34 | 1.48 |

PQR: motivación

Recordando...

- **P**: Una posición en el dataset
- **Q**: La posición obtenida a partir de aplicar el “mejor” movimiento a P, según el dataset
- **R**: Una posición aleatoria obtenida a partir de P, tal que $R \neq Q$

PQR: motivación

Recordando...

- **P**: Una posición en el dataset
- **Q**: La posición obtenida a partir de aplicar el “mejor” movimiento a P, según el dataset
- **R**: Una posición aleatoria obtenida a partir de P, tal que $R \neq Q$

Y los principios:

- 1 Si $P \rightarrow Q$, entonces $f(P) = -f(Q)$ (suma cero)
- 2 Si $P \rightarrow R$ tal que $R \neq Q$, entonces $f(R) > f(Q)$

PQR: motivación

¿Los principios funcionan en la práctica? Veamos...

PQR analysis for a network trained with target scores

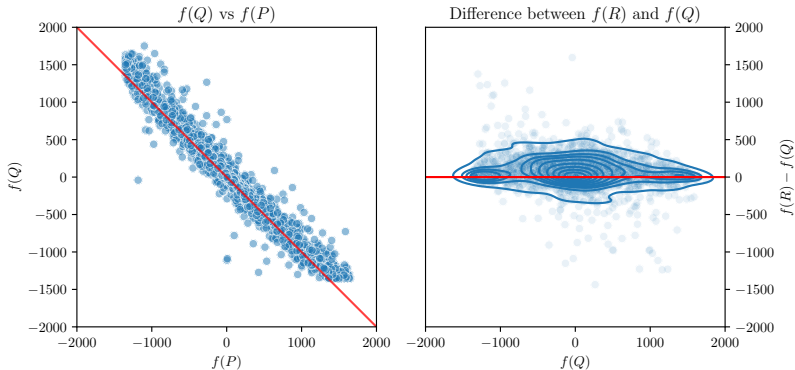


Figure: Analysis of $N = 4000$ PQR samples using a model trained with target scores and the feature set ALL.

PQR: experimento

- A. Entrenar de cero, directamente con PQR
 - no espero que sea mejor que target scores

PQR: experimento

Eligiendo R.

Conclusión

adasdas

■ asdasd