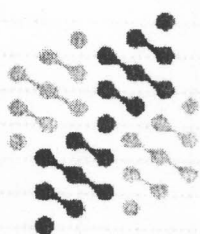


Tesis de Licenciatura

Un clasificador conexionista de posiciones de ajedrez: aprendiendo del hombre



DEPARTAMENTO
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA



Departamento de Computación
Facultad de Ciencias Exactas
Universidad de Buenos Aires

Gerardo Massenzano

L.U. 35/96

e-mail: gerardo@massenzano.com.ar

Director: Dr. Enrique Carlos Segura
Codirector: Lic. Ariel Arbiser

-- 2005 --

Indice

Resumen	4
1. Introducción	5
1.1. Motivación del presente trabajo	6
1.2. Propuesta a desarrollar	6
1.3. Procedimiento general	7
1.4. Organización de este trabajo	7
2. Estado del arte	9
3. Bases teóricas empleadas	12
3.1. Modelo conexionista	12
Aprendizaje no supervisado	12
a) Aprendizaje Hebbiano	12
b) Aprendizaje competitivo	12
3.2. Árboles de decisión	12
C4.5	13
3.3. Modelo estadístico y Bayesiano	13
Naive Bayes	13
3.4. Selección de atributos	14
FSS (feature subset selection)	14
3.5. Técnicas de estimación de la precisión del clasificador	14
Cross-validation	14
3.6. Análisis estadístico de los datos	14
Boxplot	14
4. Adquisición de datos y representación de la entrada	15
4.1. Selección de partidas	16
4.2. Selección de posiciones	16
4.3. Representación de las posiciones	17
5. El proceso de aprendizaje	19
6. Modelo de aprendizaje conexionista	20
6.1. Entrenamiento de la red de Kohonen	21
7. Experimentos y análisis de los resultados	22
Selección de atributos	22
Transformación de atributos	22
7.1. Experimentos con el modelo conexionista	23
Mejor configuración para la red de Kohonen	23
Criterios de preprocesamiento	28
7.2. Comparación del modelo propuesto con otros modelos de aprendizaje	29
7.3. Experimentos con otras bases	31
8. Conclusiones	33
Trabajo futuro	33
Agradecimientos	34
9. Bibliografía	35
Apéndice 1: Glosario ajedrecístico	37
Apéndice 2: Consultas a investigadores	38
Apéndice 3: Definición de atributos	40
Atributos	40
Definición de datos	41

Definición de predicados	41
Atributos de Peones	42
Variables usadas para las expresiones	42
Atributos de Material	44
Atributos de Movimiento	44
Atributos de Espacio	45
Atributos de Control	45
Atributos de Centro	45
Atributos de Rey	47
Atributos Especiales	49
Apéndice 4: Estadísticas de los datos utilizados	50
Boxplot	50
Matriz de correlación	53

Abstract

Experienced chess players manage to understand a position, estimating concepts of the game, without the need of the extensive expansion of moves of which the machine have to be worth themselves. Achieving the incorporation of this to a chess program is extremely difficult, mainly because of the absence of explicit rules that formalize the knowledge that humans apply. With the idea of optimizing the evaluation function of these programs, we formulate a learning model using a Kohonen self-organized map for obtaining a classifier that can decide given a game position, who was the winner of that game. A great base of games is processed, taking from them a set of positions that are described based on some attributes (like the number of pieces, the number of available moves, etc.), associated to the result of the game to which it corresponds (without draws). The network is trained with this base of classified positions for the input, generating a map of labeled neurons (white or black), that can classify new positions. Given the complexity of the problem, the experiments were carried out selecting the last position of each game. The results are compared with other two models of learning: statistical methods (Naive Bayes) and decision trees (C4.5).

Resumen

El ajedrecista experimentado logra *comprender* una posición, abstrayendo conceptos del juego, sin la necesidad de la extensa exploración de jugadas de la que se tienen que valer las máquinas. Lograr la incorporación de esto a un programa de ajedrez es sumamente complejo, en gran parte por la ausencia o desconocimiento de reglas explícitas que formalicen los conocimientos que aplica el hombre. Con la idea de perfeccionar la función de evaluación de estos programas, aquí se plantea un modelo de aprendizaje utilizando un mapa autoorganizado de Kohonen para generar un clasificador que pueda decidir, dada una posición de una partida, quién fue el ganador de la misma. Para esto se procesa una gran base de partidas, tomando de ellas un conjunto de posiciones que se describen en función de atributos (como la cantidad de piezas, la cantidad de movidas disponibles, etc.) asociados al resultado de la partida a la que corresponde (sin empates). Con esta base de posiciones clasificadas como entrada, se entrena la red generando un mapa de neuronas etiquetadas (blancas o negras), que puede clasificar nuevas posiciones. Dada la complejidad del problema, los experimentos fueron realizados seleccionando la última posición de cada partida. Los resultados se comparan con otros dos modelos de aprendizaje: estadísticos (Naive Bayes) y árboles de decisión (C4.5).

Abstract

Experienced chess players manage to understand a position, abstracting concepts of the game, without the need of the extensive exploration of moves of which the machines have to be worth themselves. Achieving the incorporation of this to a chess program is extremely complex, mainly because of the absence of explicit rules that formalize the know-how that that humans apply. With the idea of optimizing the evaluation function of these programs, we formulate a learning model using a Kohonen self-organized map for obtaining a classifier that can decide, given a game position, who was the winner of that game. A great base of games is processed, taking from them a set of positions that are described based on some attributes (like the number of pieces, the number of available moves, etc.), associated to the result of the game to which it corresponds (without draws). The network is trained with this base of classified positions for the input, generating a map of labeled neurons (white or black), that can classify new positions. Given the complexity of the problem, the experiments were carried out selecting the last position of each game. The results are compared with other two models of learning: statistical methods (Naive Bayes) and decision trees (C4.5).

1. Introducción

Los programas de ajedrez por computadora suelen dominar casi a la perfección el cálculo de fuerza bruta, esto es, la exploración exhaustiva de múltiples secuencias de jugadas posibles que se desencadenan a partir de cada posición, convirtiéndolos en muy buenos jugadores, difíciles de vencer. Ejemplo de estos programas/máquinas es el conocido Deep Blue [33], por citar al que tuvo mayor trascendencia por haberle ganado al entonces campeón mundial de ajedrez Kasparov.

Sin embargo, tienen básicamente dos limitaciones. Primero, estas secuencias de jugadas a explorar son necesariamente finitas debido al orden exponencial con el que van creciendo, lo cual deriva en un tiempo computacional muy elevado; y segundo, dado que no es computacionalmente factible explorar jugadas hasta llegar a una posición final del juego, es necesario alcanzar posiciones de cierto nivel de profundidad, y utilizar una función de evaluación que califique estas posiciones, otorgándole un valor de bondad para un jugador u otro.

Por lo general, la función de evaluación está compuesta por un conjunto de atributos cuyos valores describen la posición en otros términos más generales. Principalmente, son conceptos ajedrecísticos, como la cantidad de piezas, la cantidad de movidas disponibles, la dominación del centro, la protección de los reyes, etc. A estos atributos se los puede llamar *expertos*, ya que son formulados con conocimiento del juego. Suelen tener distintos niveles de importancia entre ellos, determinados por *pesos* que se les son asociados. De la combinación (por lo general una sumatoria) de los valores de los atributos ponderados por sus pesos, surge un nuevo valor que es el que se le asigna a la posición, para cuantificar de alguna forma la ventaja o desventaja de cada jugador. Por ejemplo, un valor positivo da ventaja a las blancas, mientras que negativo lo es para las negras.

Estas funciones, que son tan sólo una aproximación, en muchas ocasiones/su evaluación dista de ser realista. Es por esto que las *máquinas de ajedrez* todavía no han superado al hombre. Si bien es cierto que ningún ajedrecista cuenta con el poder de cálculo que tiene una computadora; el jugador experimentado la aventaja enormemente en lo que se refiere a la **comprensión** del juego en sí mismo, como ser la valoración de una posición con gran precisión, abstrayendo los conceptos posicionales, con los cuales puede determinar si una jugada es buena o no, sin necesidad de realizar cálculos de orden exponencial.

Numerosos trabajos están tomando este rumbo –apartándose un poco de la fuerza bruta por llegar a algún nivel más en la profundidad de exploración–, en la búsqueda por “aprender” del hombre la verdadera esencia del juego. En este enfoque de aprendizaje, una gran variedad de técnicas han sido aplicadas, con mayor o menor éxito. Algunos de ellos se citan en el capítulo 2.

1.1. Motivación del presente trabajo

Cuando una persona comienza a jugar por primera vez al ajedrez, tan sólo sigue las reglas del juego. Luego comienza a tener en cuenta la captura de piezas y que no le sean capturadas las propias. Con el transcurrir de las partidas, sumadas a las lecciones de libros y maestros, va mejorando su nivel de juego, incorporando nuevos "atributos". Es así que cada ajedrecista consigue tener en su mente su propia abstracción de una "función de evaluación" con la cual analiza las posiciones y determina al cabo qué jugada realizar. Pero incluso hasta el jugador más experimentado, no puede *expresar* con suficiente nivel de detalle este complejo sistema, pues hay una ausencia de reglas explícitas en la formalización de los conocimientos que aplica el jugador. Esto equivaldría en cierta medida a formular un número de reglas lógicas que sirvan para que cualquier otro jugador (esto incluye también una computadora) pueda aplicar a una posición y determine con *la mayor precisión posible* su valor de bondad. En otras palabras, que pueda decidir exactamente, dadas dos posiciones, cuál lo favorece más.

Se puede decir que el principal problema en los sistemas de ajedrez por computadora está en la evaluación de las posiciones. Ésta es una función que no se conoce y se desea aproximar. Pero curiosamente, dado que el ajedrez es un juego de *información perfecta*,[?] toda la información necesaria para evaluar una posición se encuentra presente en la distribución de las piezas en el tablero (más alguna información adicional de la partida como ser posibilidad de enroque y de peón al paso). La dificultad radica en comprender correctamente la información contenida en la posición.

Motivado por esto surge la idea de *descubrir* y *aprender* del propio hombre cómo evalúa las posiciones, formulando y/o generando conceptos posicionales, relacionándolos y estableciendo su importancia entre ellos.

1.2. Propuesta a desarrollar

El objetivo principal de este trabajo es obtener un clasificador de posiciones que pueda predecir, dada una posición de una partida, cuál fue su resultado. Se busca descubrir una representación aproximada de la expresión analítica de una función de evaluación en base a muestras de entrada-salida. Para esto se emplea un modelo de redes neuronales que es entrenado a partir de un conjunto de posiciones obtenidas de una base de partidas. Junto a ellas se utiliza el resultado de la partida correspondiente como clase.

Tomando la idea de algunos trabajos [2][27] en los que usan diferencia temporal para relacionar las posiciones de una partida, a través de una función que pondera la cantidad de jugadas que quedan hasta el final, este trabajo se basa en la siguiente hipótesis: en general, dado un jugador de buen nivel que gana una partida, las posiciones por las que transita dicha partida son en un principio equilibradas, y van favoreciendo cada vez más al ganador cuanto más se aproximen al final (posición de la partida en que la ventaja es "máxima" en relación a las anteriores). Claro que hay muchas partidas en las que hasta cierto momento son favorables para un jugador, y hasta un pequeñísimo error puede llevar a que se revierta y gane el otro. Pero supondremos la hipótesis de que si se considera un número bien grande de partidas de buenos jugadores, este error tiende a ser bajo.

La idea es lograr un aprendizaje lo más automatizado posible, evitando dar información de apreciación acerca del juego o de una posición (como por ejemplo, no contar con que una reina vale más que una torre). El aprendizaje está focalizado en posiciones del medio juego y a un alto nivel, sin entrar en consideración de análisis tácticos particulares de cada posición.

Se espera que a partir de lo desarrollado en este trabajo se pueda llegar a generar una función de evaluación para implementar en un programa de ajedrez o para extraer conceptos posicionales de interés para la comunidad ajedrecística.

1.3. Procedimiento general

El procedimiento de este trabajo consta de dos fases: la selección y representación de las posiciones, y el aprendizaje.

La primera fase consiste en la recolección y selección de los datos para el aprendizaje. Se toma una base de partidas de jugadores reconocidos, de las cuales se selecciona un conjunto de posiciones para entrenar y otras tantas para evaluar el nivel de eficiencia del aprendizaje obtenido.

Para describir las posiciones, se propone una gran cantidad de atributos *expertos*, formulados con conocimiento del juego, como número de piezas, número de movidas disponibles por ellas, dominación del centro, protección de los reyes, etc., y se asocia a cada una de ellas el resultado de la partida a la cual pertenecen. A veces las posiciones de una partida no se condicen con el resultado de la misma, ya que pasa que un jugador en el transcurso de la partida tiene ventaja, y en algún momento la puede perder por una jugada desacertada, dando la oportunidad a que el rival se quede con el triunfo. Tomando una base grande de partidas, el error de considerar una posición como ganadora, cuando realmente no lo es, se espera que sea mínimo. Para ello es importante que el método de aprendizaje sea tolerante a fallas, ya que es factible que posiciones “iguales” o muy similares, en los valores de los atributos, tengan resultados opuestos. ✓

La idea de definir nuevos atributos basados en la experiencia del aprendizaje del experto es para generalizar y reducir los atributos iniciales, es decir, todas las combinaciones de las piezas que pueden ocupar los distintos casilleros (todas las posibles configuraciones de posiciones).

Para la fase de aprendizaje, con la base de posiciones se entrena un clasificador construido a partir de un modelo de redes neuronales, un mapa autoorganizado de Kohonen. Luego, con el mapa resultante de neuronas etiquetadas, se verifica su eficiencia con nuevas posiciones.

1.4. Organización de este trabajo

En el capítulo 2 se hace una síntesis del estado del arte, referenciando a los trabajos y desarrollos recientes más destacados en el área de aprendizaje automático en el ajedrez.

En el capítulo 3 se expone una breve reseña de la teoría utilizada en este trabajo.

El capítulo 4 describe el proceso de selección de datos y representación de las posiciones.

En el capítulo 5 se presenta el proceso de aprendizaje utilizado junto con unos diagramas ilustrativos.

En el capítulo 6 se detalla el modelo propuesto de aprendizaje conexionista.

En el capítulo 7 se presenta una serie de experimentos realizados: en la sección 7.1 con el modelo propuesto; en la sección 7.2 comparándolo con otros modelos de aprendizaje automático, como árboles de decisión (algoritmo C4.5) y estadísticos (naive Bayes); y en la sección 7.3 con otras bases de posiciones. Junto con los experimentos se encuentra el análisis de los resultados.

Finalmente, en el capítulo 8 se exponen las conclusiones de este trabajo, y se plantean algunas ideas para el futuro.

2. Estado del arte

El ajedrez es un excelente y muy interesante dominio para el desarrollo de algoritmos y técnicas computacionales. En particular lo es para el área de aprendizaje automático, ya que si bien el mercado es dominado por los programas basados en exploraciones exhaustivas, un gran avance se está haciendo en el campo de la inteligencia artificial.

En dos de sus trabajos de recopilación, Johannes Fürnkranz resume muy bien acerca del rumbo que ya estaban empezando a tomar las investigaciones y trabajos en materia de ajedrez computacional: el incipiente avance del aprendizaje automático [5] y las técnicas de descubrimiento de conocimiento en bases de datos (KDD) [6]. Pero todavía hay bastante por recorrer, y son varios los caminos, ya que el aprendizaje automático cuenta con una amplia diversidad de modelos.

Focalizando principalmente desde un punto de vista más teórico, Simon y Schaeffer [25] hacen otra recopilación en la cual analizaban el juego humano, la historia del ajedrez por computadora, con sus estrategias de búsqueda versus conocimiento, y el papel que juega la teoría de juegos.

Uno de los primeros trabajos en el área de aprendizaje en ajedrez [31], trabaja hasta cierto punto con patrones y trata de hacerlo en base a un modelo del pensamiento, en vez de basarse primordialmente en búsqueda.

Gaines [7] plantea el modelo inductivo para adquirir conocimiento utilizando grafos acíclicos dirigidos de excepciones (EDAG), que son estructuras de conocimientos más compactas que los árboles y las reglas, y lo aprendido es comprensible y de utilidad para un experto ajedrecista.

Otro modelo empleado bastante es el de razonamiento basado en casos (CBR). S. Lazzeri propone un sistema interactivo de consultas de ajedrez [14] donde combina las técnicas de lógica difusa y razonamiento basado en casos para producir recomendaciones de alto nivel para posiciones específicas. Está centrado en posiciones de medio juego, y su principal propósito no es resolver problemas, sino guiar a los estudiantes de ajedrez en posiciones complejas que requieran entender conceptos estratégicos. Por otro lado, Yaakov Kerner [10] desarrolla un modelo de este tipo para proporcionar una evaluación posicional comprensible, para cualquier posición, es decir un *framework* de estrategias de aprendizaje de patrones explicativos (*explication patterns*) para la evaluación de atributos básicos del juego. Estos patrones son definidos como la configuración mínima de un pequeño número de piezas y casillas que describen sólo un atributo importante del juego.

Mediante un enfoque híbrido de aprendizaje y exploración, "Knightcap" [2], un exitoso programa entrenado jugando contra otros programas y ajedrecistas, basa su aprendizaje en una variante de diferencia temporal $-TD(\lambda)-$ y el algoritmo de búsqueda *minimax*.

En muchos casos los modelos de aprendizaje se combinan. Levinson [15] utiliza métodos combinados de aprendizaje automático, como algoritmos genéticos, actualización de pesos, $TD(\lambda)$, aprendizaje estructurado basado en reglas. Crea un

sistema de ajedrez, "Morph", con una exploración limitada de una sola movida, poco conocimiento de dominio y entrenamiento no supervisado. Mejora su juego enfrentándose a otro programa, incorporando la experiencia como pares de atributo-peso. Entre otras cosas destaca la *creatividad* de su juego.

En otro trabajo interesante, Levinson junto a Weber [16], enfatiza que la clave es representar adecuadamente las relaciones entre las piezas y evaluar las fortalezas y debilidades de ellas. En este trabajo extienden la representación de grafos, mediante grafos de vecindarios de piezas o casillas. Usan $TD(\lambda)$ y redes neuronales multicapas.

Continuando con las redes neuronales, no se puede dejar de mencionar el importante aporte de Sebastian Thrun [27] con su NeuroChess. Es un programa que aprende a jugar a partir del resultado de sus propias partidas. Utiliza una combinación de redes neuronales, $TD(\lambda)$ y una variante de aprendizaje basado en reglas, con la cual aprende su función de evaluación.

En el caso de Robertson [19], el método que utiliza es aplicar redes neuronales para clasificar un concepto estratégico e incorporarlo en una función de evaluación para una posición simple de final de juego. Usa perceptrones multicapas con backpropagation. En una de sus propuestas para trabajo futuro, propone utilizar el resultado de partidas de grandes maestros como oráculo que pueda determinar por ejemplo qué tan buena o mala es una estructura de peones. Con una idea similar, en este trabajo se incorpora el resultado de las partidas para determinar si una posición es mejor para las blancas o las negras.

Con el mismo modelo de redes, en otro trabajo realizan un sistema genético-neuronal [4] que emplea las redes como función de evaluación, y un algoritmo genético para modificar sus pesos. Las entradas a la red están dadas por diez atributos del mismo estilo de los propuestos aquí.

Respecto de otros modelos de aprendizaje automático, pero teniendo objetivos muy cercanos al presente trabajo, está el que realizan Kubat y Zizka [13]. Utilizan árboles de decisión (C5) para aprender a clasificar posiciones a través de atributos específicos, y ejemplos positivos y negativos. A diferencia de lo que se presenta aquí, el aprendizaje se aplica a posiciones específicas (sólo aquellas en las que existe un sacrificio de alfil en h7), con atributos específicos de ese conjunto de posiciones. Pero el enfoque es bastante similar: un conjunto de posiciones representadas por estos atributos y su clasificación (si el sacrificio es positivo o no). La eficiencia que logran es muy buena (87.7%). Las posiciones son tomadas de una base de datos comercial, luego seleccionan las que cumplen el criterio descrito, y usan un programa comercial para determinar si es correcto el sacrificio para asignar así la clase correspondiente. Usan atributos triviales (la ubicación de las piezas en el tablero) y atributos expertos, como en el presente trabajo (aunque los aquí propuestos son más generales, como se detalla en la sección 4.3 y en el apéndice 3).

Zizka y Madr [30] profundizan este último trabajo, utilizando 8 formas distintas para representar las posiciones. Dos de ellas generando automáticamente los atributos, otras dos con atributos expertos, y las 4 restantes una combinación de los dos métodos, obteniendo el mejor resultado de esta forma. El número de ejemplos incide en la

eficiencia de la clasificación: cuanto más son, mejor es ésta. Algo que sorprende es que usando sólo 64 atributos generados automáticamente obtuvieron buenos resultados, alrededor del 10% de error con *boosting*.

Se podría seguir citando más trabajos pues, como se ha visto por los ya mencionados, se exploran distintas alternativas, con objetivos similares: básicamente, lograr el aprendizaje de la función de evaluación. El tema está en plena etapa de desarrollo, por la complejidad del problema. Poco hay elaborado acerca del objetivo apuntado por este trabajo, y justamente se debe a la dificultad del mismo. Por esto es que varios de estos autores fueron consultados, para que dieran sus apreciaciones acerca de este trabajo y compartan sus ideas y conocimiento al respecto. Sus respuestas fueron diversas, pero todos coincidieron en que el tema abordado aquí es sumamente complejo. En el apéndice 2 se sintetiza este valioso aporte.

3. Bases teóricas empleadas

Aquí se ofrece una breve introducción de los métodos y modelos de aprendizaje utilizados para obtener el clasificador de posiciones, así como algoritmos de selección y transformación de atributos para el preprocesamiento de la entrada.

3.1. Modelo conexionista

Los métodos de aprendizaje con redes neuronales artificiales están inspirados en parte en la observación de los sistemas de aprendizaje biológicos, que poseen una compleja red de neuronas interconectadas, donde a partir de una entrada se genera un resultado. Proveen un método práctico y general para el aprendizaje a partir de ejemplos de funciones reales, discretas e intervalares. Este aprendizaje es robusto frente a la aparición de errores en los datos de entrenamiento [8].

Aprendizaje no supervisado

La red basada en este principio no tiene retroalimentación del entorno, debe descubrir por sí misma sus patrones, características, correlaciones o categorías de la entrada. Estas redes son útiles cuando hay redundancia en los datos de entrada.

a) Aprendizaje Hebbiano

Es una técnica de aprendizaje no supervisado, basada en conexiones, donde las unidades de salida no compiten entre sí, sino que se activan simultáneamente. En este caso, para el aprendizaje de la red se utiliza la regla de Sanger [22] (una modificación a la regla de Hebb [9]), que permite obtener las componentes principales. Despreciando las componentes menos significativas, se puede reducir la dimensión de la entrada. Es decir que se puede transformar la entrada con poca pérdida de información.

b) Aprendizaje competitivo

En este caso, las unidades de salida compiten entre sí por ser activadas, aunque también de forma no supervisada. El objetivo de estas redes es agrupar o categorizar los datos de entrada. Similares entradas deberán ser clasificadas en la misma categoría, y por lo tanto debería activarse la misma unidad de salida. Las clases deben ser descubiertas por la red misma, a partir de las correlaciones de los datos de entrada.

Algoritmo de Kohonen [12]: es una variante en la cual las neuronas vecinas a la ganadora se acercan a ella, proporcionalmente en función a su respectiva *distancia*. El resultado obtenido es un mapa topográfico, que preserva las relaciones de vecindad, y permite visualizar la entrada reflejada en la distribución de las unidades de salida en un espacio de una o dos dimensiones (esta última es la más utilizada).

3.2. Árboles de decisión

Es un método para la aproximación de funciones de valores discretos, robusto frente a datos con ruido y capaz de aprender expresiones disjuntas. La función aprendida es

representada por un árbol de decisión. Estos métodos buscan un espacio de hipótesis completamente expresivo y de esta manera evitan las dificultades que surgen de espacios de hipótesis restringidos. Su sesgo inductivo es la preferencia de árboles pequeños por sobre los grandes. Estos árboles clasifican las instancias ordenándolas desde la raíz hacia algún nodo hoja, proveyendo la clasificación de la instancia. Cada nodo del árbol especifica la evaluación de un algún atributo de la instancia y cada rama que baja desde dicho nodo se corresponde con cada valor que puede tomar el atributo. Estos árboles permiten ser expresados como un conjunto de reglas lógicas, posibilitando al hombre una mejor comprensión del aprendizaje realizado [17].

C4.5

Es un algoritmo de la familia de árboles de decisión, surgido como mejora del primitivo ID3 [20]. Básicamente incorpora la poda de ramas para evitar el sobreaprendizaje de las instancias de entrenamiento, posibilitando una mejor generalización [21].

3.3. Modelo estadístico y Bayesiano

El razonamiento Bayesiano provee un acercamiento probabilístico a la inferencia. Está basado en asumir que las cantidades de interés están gobernadas por distribuciones de probabilidad y que las decisiones óptimas pueden ser realizadas por medio de razonamientos sobre estas probabilidades y datos observables. Provee una visión cuantitativa para pesar la evidencia que soporta distintas hipótesis. El razonamiento Bayesiano provee las bases para algoritmos de aprendizaje que manipulan directamente probabilidades, y un ámbito para analizar como operan otros algoritmos que no las manipulan explícitamente [17].

Naive Bayes

Es un clasificador del modelo Bayesiano que se usa para tareas de aprendizaje donde cada instancia x es descripta por una conjunción de valores de atributos y cuya función a aprender $f(x)$ puede tomar cualquier valor de un conjunto finito V . Este clasificador está basado en asumir por simplicidad que los valores de atributo son condicionalmente independientes de la clase. En otras palabras, se asume que dada la clase de una instancia, la probabilidad de observar la conjunción $a_1, a_2, \dots, a_n$ es el producto de las probabilidades de cada atributo. No se realiza una búsqueda explícita sobre el espacio de posibles hipótesis. En su lugar, la hipótesis está formada simplemente contando la frecuencia de las varias combinaciones de los en los ejemplos de entrenamiento.

3.4. Selección de atributos

Es el proceso de identificar y descartar la información irrelevante y redundante tanto como se pueda. Esto reduce la dimensionalidad de los datos y permite a los algoritmos de aprendizaje desempeñarse más rápida y efectivamente. En algunos casos, el error de generalización puede mejorar, y en otros el resultado es una forma más compacta y fácil de interpretar del concepto a aprender.

FSS (feature subset selection) → ¿es el nombre real?

Es una familia de algoritmos de clasificación que se valen de otro clasificador para seleccionar un buen subconjunto de atributos que mejoran su eficiencia. Uno de ellos (utilizado aquí), va iterando en el número de atributos elegidos, empezando desde subconjuntos de un atributo, incorporando el mejor atributo para el clasificador utilizado, hasta que al agregar un atributo más la clasificación no mejora [11]. Para los experimentos de este trabajo se utilizó naive Bayes como clasificador de FSS.

3.5. Técnicas de estimación de la precisión del clasificador

Cross-validation

Es una técnica estadística que provee una estimación de la precisión del clasificador representativa de todos los datos utilizados. Consiste en tomar aleatoriamente instancias de todo el conjunto, y separarlas en N particiones u hojas (folds) de aproximadamente la misma cantidad. Cada una de ellas es utilizada para evaluar, mientras que las restantes ($N-1$ particiones) se usan para entrenar el clasificador. Esto se repite para cada partición (realizando el procedimiento N veces). Luego se toma como error del clasificador el promedio de las N evaluaciones realizadas. La forma estándar de predecir el promedio de error de una técnica de aprendizaje, dado un conjunto de datos único y prefijado, es realizar cross-validation de 10 hojas (*tenfold cross-validation*) [29].

3.6. Análisis estadístico de los datos

Boxplot

Es un tipo de gráfico que es utilizado para mostrar la forma de la distribución, su valor central, y variabilidad. La figura producida consiste en los valores más extremos del conjunto de datos (valores mínimo y máximo), la mediana, y los cuartiles inferior y superior (medianas calculadas entre la mediana y los ambos extremos, respectivamente). Resulta una forma eficiente de resumir un conjunto de información distribuida en un intervalo. Es usado comúnmente en análisis exploratorio de los datos [35]. Aquí se utiliza para graficar la entrada para cada uno de los atributos utilizados.

4. Adquisición de datos y representación de la entrada

La base de partidas fue obtenida de un sitio público de Internet [32]. La base completa cuenta con alrededor de 500.000 partidas de grandes maestros y ajedrecistas de alto nivel en general. Están en formato PGN, y fueron jugadas entre los años 1970 y 2002.

Con un conjunto de aproximadamente 18.000 posiciones de la base mediojuego final, correspondientes cada una a la última posición de su partida (ver sección 4.2), se entrenó una red Hebbiana con la regla de Sanger, proyectando así los 65 atributos iniciales y seleccionando luego las primeras dos componentes principales. Con ello se puede graficar la distribución de la entrada en dos dimensiones, distinguiéndola de acuerdo a su clase, es decir, el resultado de la partida (figura 4.4.1).

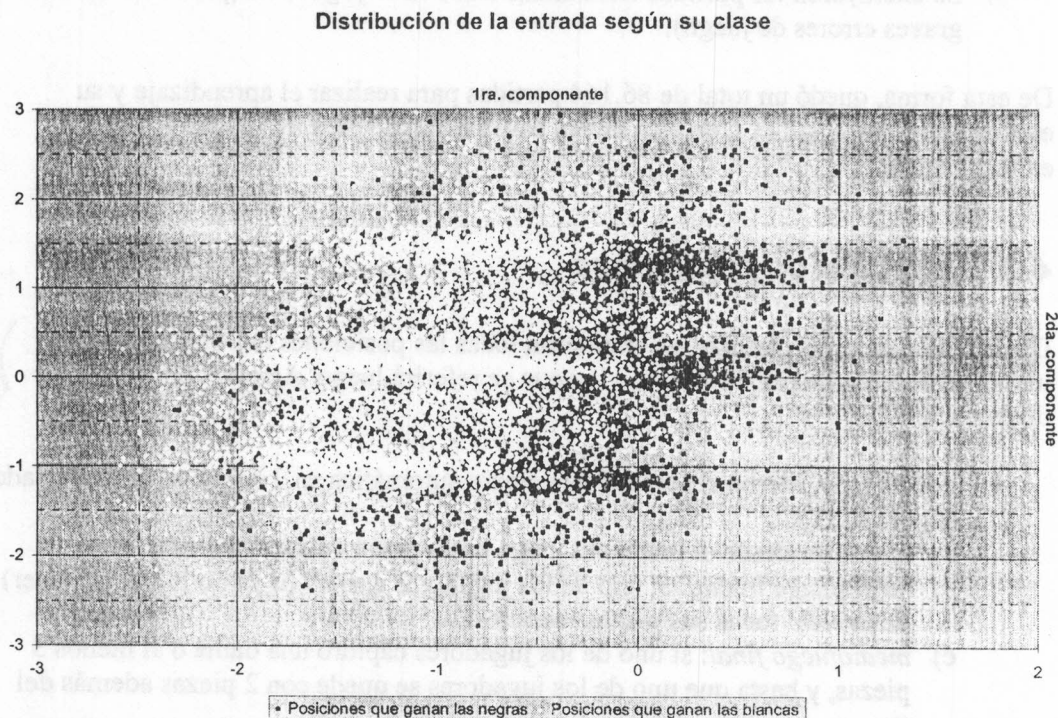


Figura 4.4.1. Distribución de las posiciones proyectando sus atributos en las primeras dos componentes principales. Según el ganador de la partida correspondiente, se visualizan en color blanco o negro.

Del gráfico de la figura 4.4.1 se desprende que hay un alto grado de solapamiento de las clases cuando las instancias son proyectadas sólo por sus primeras dos componentes principales. En otras palabras, sucede que en muchas posiciones, son iguales o muy similares en los valores de los atributos y el ganador de la partida es tanto el jugador de negras como el de blancas. Esta es una muestra muy interesante para poder apreciar la complejidad del problema, donde la parte más crítica y difícil está en la adecuada descripción de las posiciones.

4.1. Selección de partidas

Con la ayuda de un programa para manejo de bases de partidas [34], se realizó, en primer lugar, una depuración de partidas que tuvieran algún error, o que estuvieran duplicadas. Luego de esto, se tomaron varias decisiones de selección:

1. Fueron descartadas las partidas cuyo resultado fuera empate (para simplificar el aprendizaje: gana el blanco o el negro).
2. Se seleccionaron las partidas en las que ambos jugadores tuvieran un ranking [ELO] superior a 2300 (para asegurarse que las partidas a utilizar fueran de un buen nivel). Partidas en las que uno o ambos jugadores carecían de esta información tampoco fueron consideradas.
3. Se excluyeron las partidas terminadas antes de la jugada 15 (procurando evitar graves errores de juego).

De esta forma, quedó un total de 86.164 partidas para realizar el aprendizaje y su evaluación (figura 4.2.1). Con esta nueva base se trabajó para los distintos experimentos.

4.2. Selección de posiciones

De cada una de estas partidas, se extrajeron todas las posiciones del transcurso del juego, representadas por atributos, a los que se referirá luego. Aquí se tomaron otras decisiones:

¿después?

1. Por un lado, fueron separadas de manera automática en 3 bases, según su estado (figura 4.2.1):
 - a) *apertura*: desde el inicio del juego hasta pasar a algún otro estado;
 - b) *mediojuego inicial*: si uno de los jugadores *enrocó* (o ya no lo puede hacer) y *desarrolló* todas sus piezas menores;
 - c) *mediojuego final*: si uno de los jugadores capturó una dama o al menos 3 piezas, y hasta que uno de los jugadores se quede con 2 piezas además del rey

El motivo de esta separación es porque se asume que en los distintos pasajes de la partida se tienen en consideración distintos atributos (o en otros términos, varía el peso de importancia de cada atributo), y de esta forma se puede facilitar el aprendizaje. La fase final de la partida no fue considerada para este trabajo, ya que el enfoque está puesto principalmente en el mediojuego. La división propuesta es arbitraria.

2. Se excluyeron las posiciones "inseguras", consideradas así (en términos generales) a las que se encuentran en medio de una secuencia de intercambio de piezas, o en las que uno de los reyes esté en jaque. Esto es para evitar aprendizajes incorrectos (por ejemplo, ficticios desbalances de material, o poca movilidad por estar en jaque).

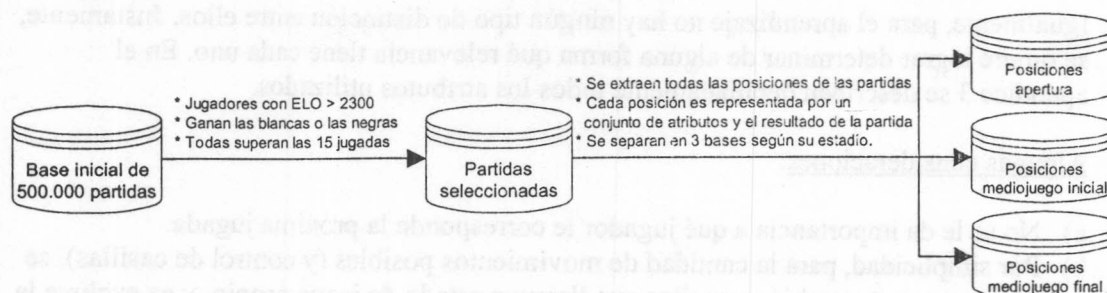


Figura 4.2.1. Proceso de selección de partidas y de posiciones utilizadas para los experimentos.

Dada la complejidad del problema abordado, las posiciones utilizadas para los experimentos son las últimas de cada partida, donde la ventaja del ganador es más evidente (asumiendo la hipótesis enunciada en la sección 1.2).

4.3. Representación de las posiciones

Cada posición está representada por un conjunto de atributos que la describe, junto con el resultado de la partida a la que corresponde, que se utilizará para ser clasificada. Como fue mencionado en la introducción, éste es el punto más crítico ya que aún se desconocen mejores representaciones realistas. En muchos casos, en la literatura existen conceptos de atributos, generalmente a un nivel bastante abstracto, y rara vez se especifican con predicados lógicos, necesarios para poder definirlos sin ambigüedad y obtener sus valores.

Estos atributos fueron definidos y/o seleccionados por el autor basándose en su propia experiencia, y tomando además algunas ideas de libros de ajedrez, publicaciones en Internet, funciones de evaluación de varios programas¹, y la asistencia de algunos importantes jugadores de ajedrez². Para disponer de la mayor información posible, se definió una buena cantidad de atributos, con la idea de descartar los menos relevantes. Los atributos propuestos pueden ser categorizados de la siguiente manera:

Material: cantidad por cada tipo de pieza.

Movimiento: tiempo, desarrollo, movilidad (total y por pieza, y por sectores del tablero).

Espacio: control de espacio, ventaja de espacio.

Control de casillas: control absoluto, ventaja de control, control mínimo (en total y por sector del tablero).

Centro: ocupación, centralización de las piezas.

Peones: mayorías de flanco, cantidad de islas, aislados, doblados, avanzados, pasados, colgantes.

Rey: centralización, defensa de peones, cercanía de piezas propias y adversarias, control de casillas próximas al rey opuesto.

Otros: torres en columnas abiertas, en séptima fila, conectadas, alfiles y caballos en casillas fuertes.

¹ KnightCap, Crafty, DeepThought, entre otros.

² GM Oscar Panno, MI Ernesto Juliá, Juan Pablo Panebianco.

Igualmente, para el aprendizaje no hay ningún tipo de distinción entre ellos. Justamente, se quiere lograr determinar de alguna forma qué relevancia tiene cada uno. En el apéndice 3 se describen detalladamente todos los atributos utilizados.

Algunas consideraciones:

- No se le da importancia a qué jugador le corresponde la próxima jugada.
- Por simplicidad, para la cantidad de movimientos posibles (y control de casillas), se toma en cuenta también aquellos que lleven a estado de jaque propio, y se excluye la disponibilidad de peón al paso.
- Todos los atributos se expresan con un valor numérico discreto, y la mayoría de ellos reflejan la diferencia de un jugador sobre el otro (por Ej. el valor del atributo movilidad, es la diferencia entre los movimientos disponibles por el blanco con los del negro, quedando un valor positivo si la ventaja es del primero, y negativo para el segundo).

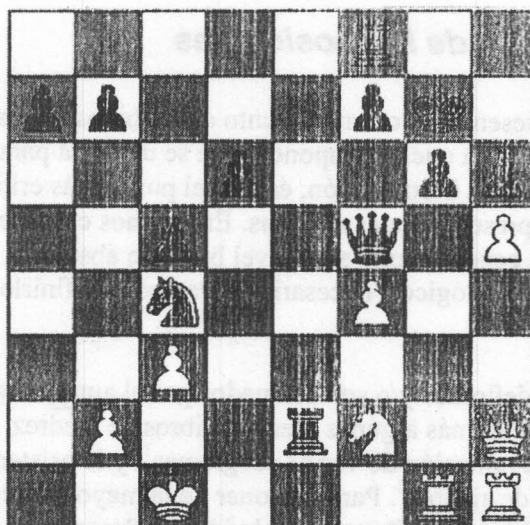


Figura 4.3.1. Ejemplo de posición utilizada para el aprendizaje.

En la figura 4.3.1 se muestra el diagrama de una posición de las utilizadas en los experimentos. La tabla 4.3.1 corresponde a los valores para esta posición de algunos de los atributos propuestos. Todos ellos están expresados en el valor para las blancas menos el respectivo para las negras.

Atributo	Valor
Nro. de peones	-3
Nro. de caballos	0
Nro. de alfiles de casillas blancas	0
Nro. de alfiles de casillas negras	0
Nro. de torres	0
Nro. de reinas	0
Nro de movidas de peones	-5
Nro de movidas de caballos	-2
Nro de movidas de alfiles	0

Atributo	Valor
Nro de movidas de torres	-8
Nro de movidas de reina	-11
Nro de movidas de rey	-3
Nro. de movidas	-29
Torres en fila 7	-1
Torres conectadas	1
Islas de peones	1
Nro. de peones aislados en columna cerrada	2
Peones avanzados	1

Tabla 4.3.1. Ejemplo de pares de atributo y valor para una posición.

5. El proceso de aprendizaje

El problema está orientado a un enfoque de aprendizaje, ya que lo que se quiere es aproximar la expresión analítica de una función de evaluación, la cual es desconocida. Se busca generar una representación aproximada en base a muestras de entrada-salida. El proceso de aprendizaje utilizado en los distintos experimentos para obtener el clasificador de posiciones, cuenta con dos fases generales:

1. La primera consiste en preprocesar la entrada, esto es, al conjunto de posiciones representadas por los atributos mencionados se le aplica una transformación, una selección de los mejores atributos (descartando los más irrelevantes), o una combinación de estas dos. Otra opción es realizar directamente la siguiente fase sin ningún preprocesamiento de la entrada (figura 5.1).

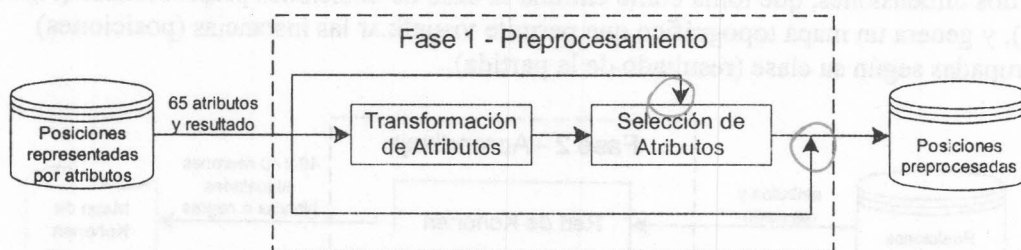


Figura 5.1. Fase de preprocesamiento de las posiciones.

2. Utilización de un modelo de aprendizaje para la construcción del clasificador, a partir de las posiciones preprocesadas (figura 5.2).

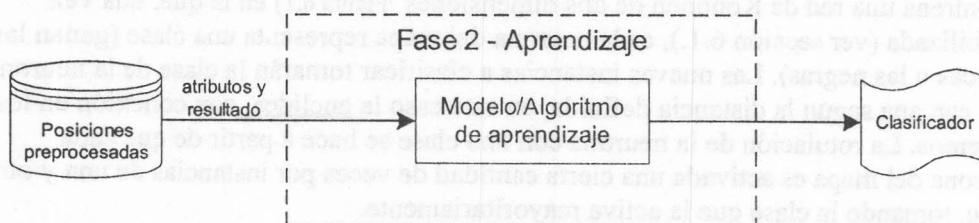


Figura 5.2. Fase de aprendizaje.

Por último, para medir la eficiencia del clasificador, se lo evalúa con un conjunto de posiciones no usadas para el entrenamiento.

6. Modelo de aprendizaje conexionista

Los primeros experimentos en este modelo para la fase de aprendizaje fueron con un enfoque supervisado, utilizando los clásicos perceptrones multicapas con backpropagation [8]. Pero ya en la etapa de aprendizaje, las redes no dejaban de oscilar, y no se pudo converger a un resultado. Es muy posible que esto se deba a la particularidad de los datos, ya que posiciones muy similares, tal vez con el valor de sólo un atributo distinto, correspondan a partidas de resultados opuestos. Esto hace que la función de evaluación sea intrínsecamente inestable, por lo cual es difícil poder reflejarla en un modelo esencialmente continuo, en el cual las redes tienen poca sensibilidad de la salida a pequeñas variaciones de la entrada.

Finalmente se trabajó con una arquitectura de red no supervisada. Una red de Kohonen de dos dimensiones, que toma como entrada la base de posiciones preprocesadas (Figura 5.1), y genera un mapa topográfico que permite visualizar las instancias (posiciones) agrupadas según su clase (resultado de la partida).

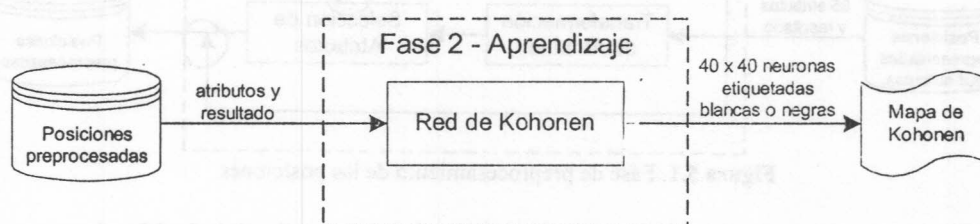


Figura 6.1. Fase de aprendizaje utilizando la red de Kohonen como modelo.

Se entrena una red de Kohonen de dos dimensiones (Figura 6.1) en la que, una vez estabilizada (ver sección 6.1.), cada neurona del mapa representa una clase (ganan las blancas o las negras). Las nuevas instancias a clasificar tomarán la clase de la neurona más cercana según la distancia definida; en este caso la euclídea, con conexión en los extremos. La rotulación de la neurona con una clase se hace a partir de que cada neurona del mapa es activada una cierta cantidad de veces por instancias de una y otra clase, tomando la clase que la activa mayoritariamente.

Para que la clasificación sea buena, se espera que esta proporción de activación sea lo más favorable para una de las dos clases en cada neurona. Con lo cual, teniendo en cuenta que luego habrá que evaluar con nuevos datos, se busca generalizar la clasificación escogiendo un umbral para descartar las neuronas que tienen mayor error de clasificación (de las instancias de entrada). Por Ej., si el umbral es .75 una neurona que haya clasificado 60 posiciones como ganadoras a las blancas y 40 a las negras, quedaría descartada (.6); mientras que otra neurona que clasificó ganadora 10 veces a las blancas y 40 a las negras (.8), será utilizada para la evaluación del clasificador.

Una vez ya con las neuronas seleccionadas, se evalúa la bondad del clasificador. Se preprocesan todas las instancias de evaluación (fase 1), y la salida se proyecta en el mapa de neuronas construido por la red de Kohonen. Dado que las instancias de prueba tienen también su clase, ésta se compara con la de la neurona activada, contando así la cantidad de aciertos y errores.

6.1. Entrenamiento de la red de Kohonen

El entrenamiento de esta red se divide en épocas. En cada época se ordenan aleatoriamente las instancias de entrenamiento y se procesan una a una para que la red aprenda, corrigiendo los pesos de acuerdo a la neurona que se activa con cada instancia. Cada n épocas (5 en los experimentos) se obtiene el error de clasificación de la red a partir de las instancias de evaluación. Cuando este error se estabiliza o empieza a aumentar en algún muestreo, síntoma de sobreaprendizaje, y que la red empeora la generalización, se da por finalizado el entrenamiento de la red.

7. Experimentos y análisis de los resultados

Para los experimentos se utilizaron 65 atributos, con valores discretos como se menciona en el capítulo 4. Estos valores fueron normalizados a $[-1, 1]$. A estos atributos se les asocia la clase, donde un valor 1 equivale a un triunfo de las blancas, mientras que -1 lo es para las negras.

El entrenamiento es con 2/3 de las instancias, dejando 1/3 para la evaluación de eficiencia del clasificador. Para estimar la precisión del clasificador, se utiliza la técnica de cross-validation con 10 hojas. En todos los casos, la proporción de instancias con cada clase es siempre la misma (50%).

El conjunto de posiciones utilizado en los experimentos (de las secciones 7.1 y 7.2) corresponde sólo a la última posición de cada partida de la base mediojuego final (detallado en 4.2.). Totalizan 31.550 posiciones, utilizando 21.034 para el entrenamiento y 10.516 para la evaluación.

Junto con los experimentos se analizan los resultados obtenidos.

Selección de atributos

Se consideraron 4 criterios para seleccionar atributos:

- S1. Varianza de cada atributo para toda la muestra.
- S2. Diferencia de valores medios de cada clase.
- S3. Diferencia de valores medios de cada clase menos la suma de sus varianzas.
- S4. Algoritmo de selección FSS, utilizando como clasificador a naïve Bayes.

Para realizar la selección, se ordenan los atributos de forma decreciente según el criterio elegido, y se eligen los primeros atributos.

Transformación de atributos

Se decidió experimentar también con una transformación previa de la entrada. Para esto se construyó una red Hebbiana con la regla de Sanger para el aprendizaje, permitiendo obtener las componentes principales y a su vez reducir la dimensión de la entrada.

7.1. Experimentos con el modelo conexionista

Dado el alto costo computacional que tiene el entrenamiento de las redes neuronales, se decidió realizar un preprocesamiento de la entrada para poder reducir los 65 atributos iniciales de manera de poder acelerar el tiempo de aprendizaje. Para esto se eligió utilizar el algoritmo de selección de atributos FSS, usando como clasificador a Naive Bayes. De esta forma fueron seleccionados tan sólo 16 atributos que, como se verá en los experimentos, incluso resultó ser claramente mejor que usar todos los atributos. Esos 16 atributos son: A1, A5, A6, A7, A9, A20, A21, A22, A23, A24, A26, A34, A46, A47, A55, A60 (para su correspondencia ver apéndice 3).

→ tiene sentido la selección?

Mejor configuración para la red de Kohonen

Luego de realizar diversos experimentos variando los distintos parámetros que configuran la red de Kohonen, se pudo llegar a la mejor combinación de ellos. La dimensión del mapa de salida con la que mejor resultado se obtuvo es de 40 x 40 neuronas. En la figura 7.1.1 se comparan los resultados obtenidos con mapas de distintas dimensiones. Luego, en la figura 7.1.2 se prueban distintos umbrales de activación, que descartan para la evaluación a todas las neuronas cuyo porcentaje de activación de la clase ganadora en el entrenamiento estuviera por debajo del porcentaje elegido como umbral. Aquí la mejor opción fue 80%. También quedaron establecidos los valores de otros parámetros, como ser: la velocidad de aprendizaje inicial en 0.8; el coeficiente inicial que modula el radio del vecindario en 0.95; y el factor de reducción en cada época de estos dos parámetros en 0.8. En la tabla 7.1.1 se presenta la mejor configuración obtenida para la red de Kohonen.

Parámetro	Valor
Dimensión mapa	40 x 40 neuronas.
Umbral de activación	Al menos 80% de activación de las clases ganadora para cada neurona del mapa.
Velocidad de aprendizaje inicial	0.8
Coeficiente inicial que modula el radio del vecindario	0.95
Factor de reducción	0.8

Tabla 7.1.1. Mejor configuración de la red de Kohonen.

En la figura 7.1.1 se puede apreciar un conjunto de experimentos en donde lo que varía es el número de neuronas del mapa de Kohonen. Se observa claramente cómo el mapa de 40x40 neuronas es donde se obtuvo el menor error de clasificación. Esto indica que los mapas de dimensiones más pequeñas tienen menos poder de división de las instancias, quedando ellas distribuidas en menos neuronas y, por consiguiente, las dos clases muy acopladas. Por otro lado, los mapas de mayores dimensiones tienden al sobreentrenamiento, perdiendo poder de generalización, llegando al extremo cuando hay una neurona distinta para cada instancia de entrenamiento.

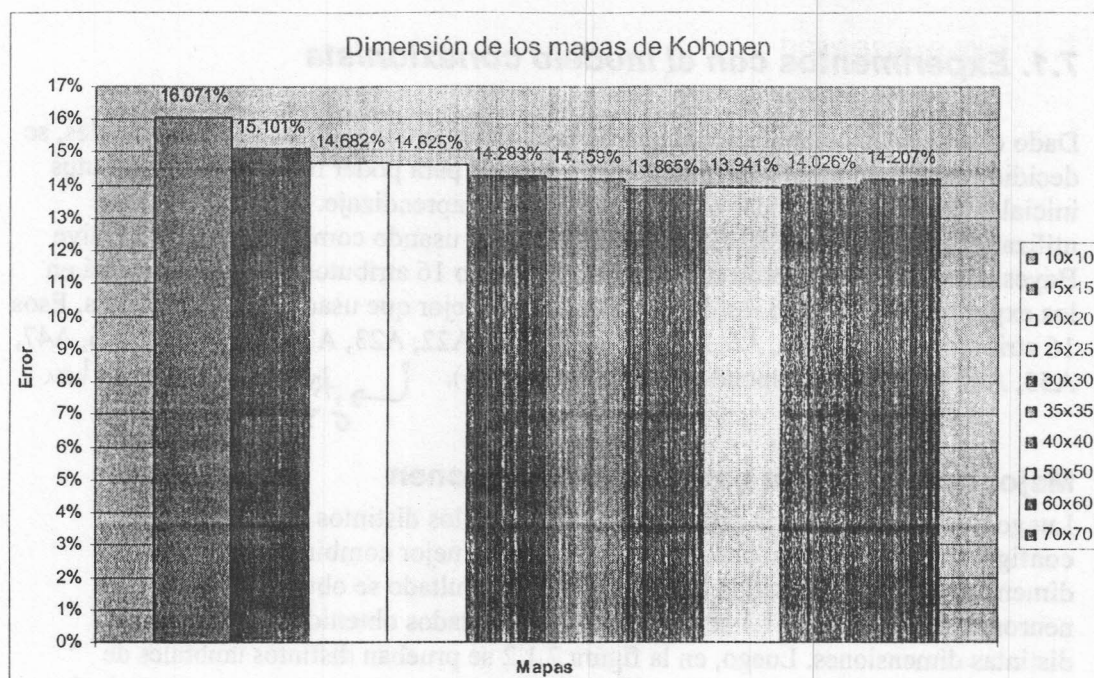


Figura 7.1.1. Comparación entre mapas de Kohonen de distinta dimensión.

Como fue explicado en el capítulo 6, luego del entrenamiento de la red de Kohonen, se busca generalizar la clasificación escogiendo un umbral para descartar las neuronas que tienen mayor error de clasificación. En la figura 7.1.2 se muestra una comparación del error de clasificación de una red ya entrenada, evaluándola con distintos umbrales.

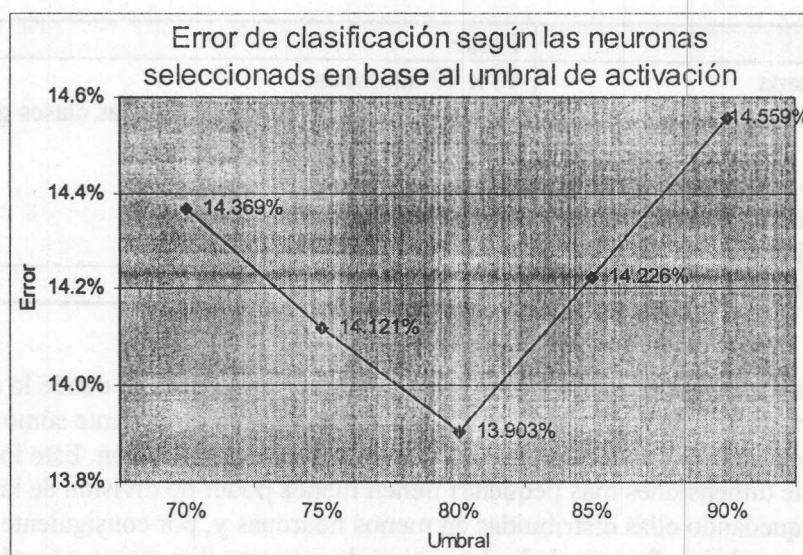


Figura 7.1.2. Variación del error según el umbral del porcentaje mínimo de activación de la clase ganadora para cada una de las neuronas del mapa.

El extremo de máxima entropía es cuando no hay umbral, que equivale a un umbral del 50%, donde no se descarta ninguna neurona. Esto ocurre porque se toman en cuenta

neuronas que aportan confusión, cuyo error de clasificación individual es muy alto. En el otro extremo, si se toma un umbral cercano al 100%, quedan seleccionadas muy pocas neuronas (o ninguna), perdiendo el poder de discriminación, ya que todas las instancias a evaluar van a ser activadas por estas pocas neuronas. En el gráfico se ve con claridad que donde se obtuvo el menor error de clasificación fue utilizando un umbral de 80%.

La figura 7.1.3 muestra un interesante gráfico con el mapa de neuronas etiquetadas, obtenido luego del entrenamiento de una red de Kohonen utilizando la configuración descrita en la tabla 7.1.1. Las neuronas contiguas etiquetadas con la misma clase forman *clusters*. Dado el umbral elegido, hay neuronas que quedan descartadas para la clasificación (se ven en color gris).

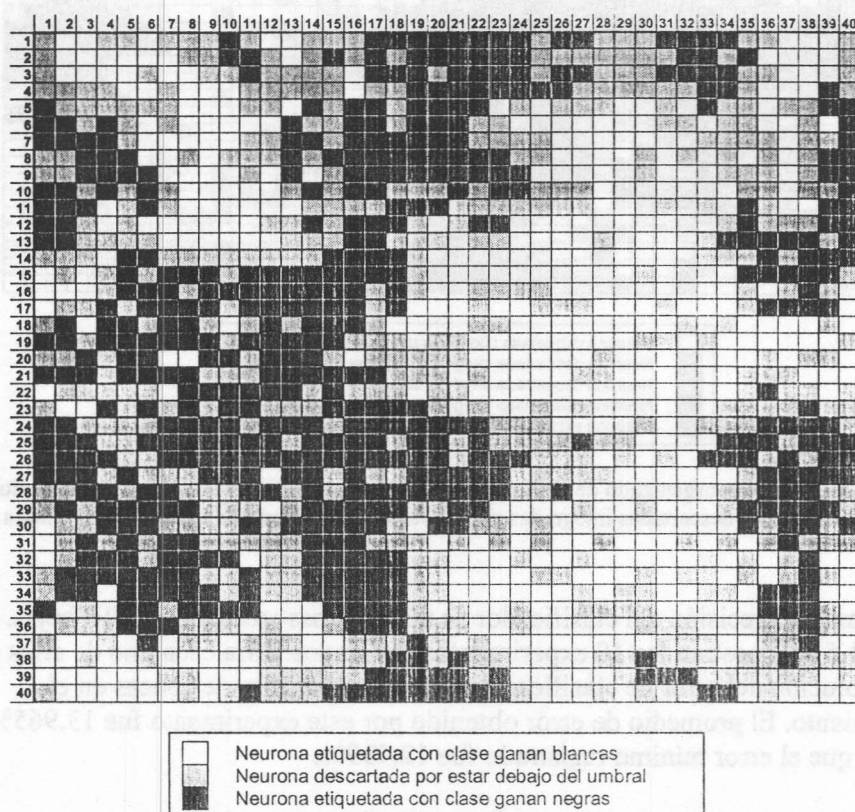


Figura 7.1.3. Mapa autoorganizado de Kohonen con las neuronas etiquetadas con una de las dos clases, incluyendo las neuronas descartadas para la clasificación por estar debajo del umbral.

Se puede considerar que las instancias que activarían las neuronas descartadas que están más cercanas a un *cluster* de una de las clases como si formaran parte de él, ya que finalmente una de sus neuronas será activada. De esta forma, en la figura 7.1.4 se muestra el mismo mapa, coloreando las neuronas descartadas que se pueden considerar integradas a algún *cluster*. Las neuronas que están en el borde de dos *clusters*¹ permanecen indefinidas, ya que las instancias que la activarían serán repartidas entre ambos.

¹ La distancia entre dos neuronas es la diferencia de sus pesos correspondientes, y no su ubicación en el mapa.

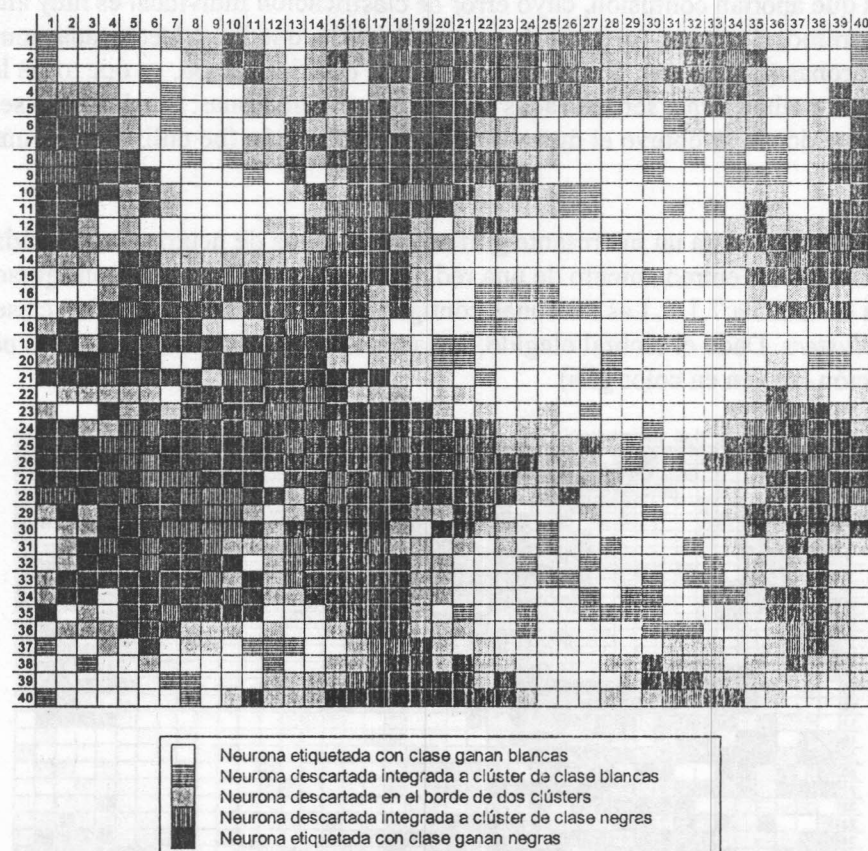


Figura 7.1.4. Mapa autoorganizado de Kohonen con las neuronas etiquetadas con una de las dos clases, y las neuronas descartadas integradas por su cercanía a clusters de algunas de las clases.

Para estimar la precisión del clasificador se utiliza cross-validation de 10 hojas. En la figura 7.1.5 se muestran los 10 experimentos realizados. Para cada uno de ellos se puede ver la evolución del error de clasificación con el incremento de épocas en el entrenamiento. El promedio de error obtenido por este experimento fue 13.965%, mientras que el error mínimo registrado fue 12.999%.

Por un lado, se puede ver que para cada par de subconjuntos de entrenamiento y evaluación, los resultados están con cierta dispersión entre 13 y 15%, aproximadamente. A su vez, en casi todos ocurre que el valor mínimo lo alcanzan tan sólo después de 5 o 10 épocas de entrenamiento. Lo más importante en relación a este experimento es que se puede decir que el par de subconjuntos de entrenamiento y evaluación utilizados para los demás experimentos es representativo, ya que el promedio de error alcanzado con cross-validation está a lo sumo apenas un 0,1% por encima del obtenido con esos subconjuntos.

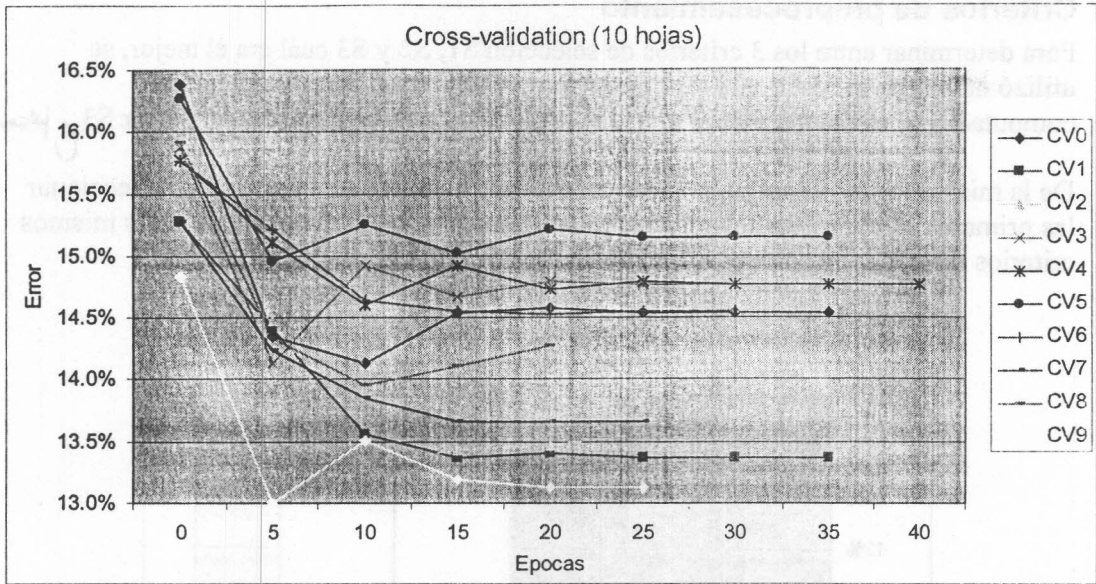


Figura 7.1.5. Cross-validation de 10 hojas.

Se evaluó como se comporta el aprendizaje de la red en relación a la cantidad de instancias de entrenamiento. En la figura 7.1.6 se muestran 4 experimentos en los que para cada entrenamiento se fue agregando de a 5.000 instancias. Para evaluar siempre se utilizó el mismo conjunto de 10.516 instancias ya seleccionadas para los experimentos previos.

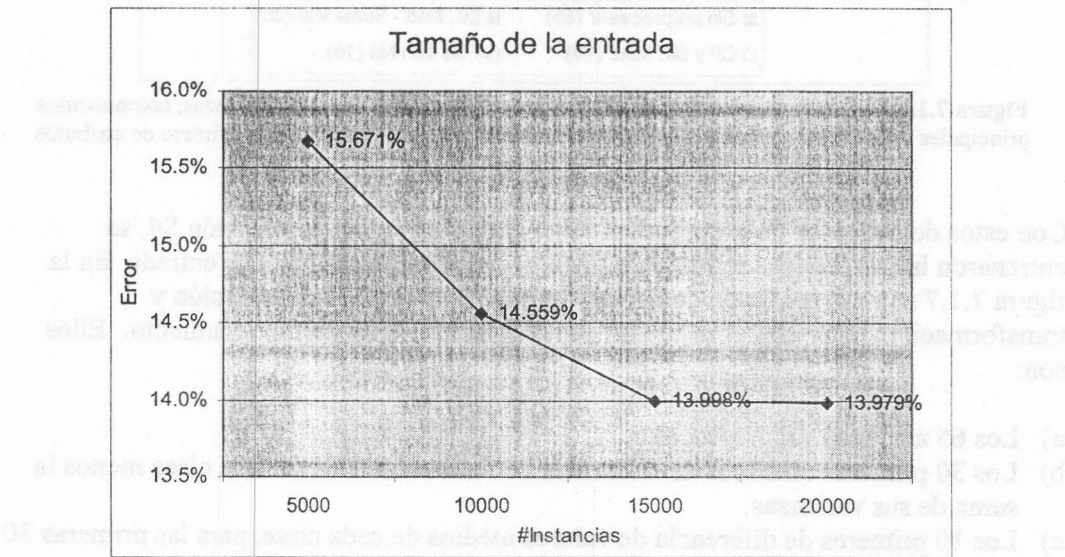


Figura 7.1.6. Evolución del error con el incremento de la cantidad de instancias de entrenamiento.

El error decrece rápidamente para luego ir estabilizándose, cercano al número de instancias (21.034) utilizado para el entrenamiento de la mayoría de los experimentos. Con esto se puede afirmar que hay una buena representatividad de los datos en los subconjuntos elegidos para aprender y evaluar.

Criterios de preprocesamiento

Para determinar entre los 3 criterios de selección S1, S2 y S3 cuál era el mejor, se utilizó naive Bayes, dado que resultó ser muy confiable, y porque su tiempo computacional es mucho menor al de las redes neuronales. El mejor resultó ser S3.

De la misma forma, se aplicó la transformación para proyectar la entrada y seleccionar las primeras 30 componentes principales, utilizando luego naive Bayes con los mismos criterios de selección, siendo el mejor en este caso S2.

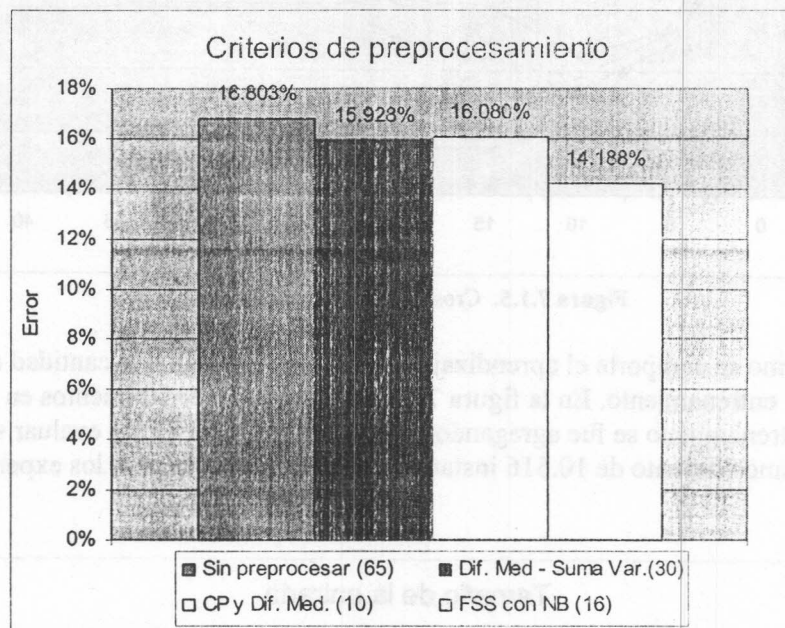


Figura 7.1.7. Criterios de selección: diferencia de medias menos suma de varianzas, componentes principales y diferencia de medias, y FSS con naive Bayes. Entre paréntesis el número de atributos utilizados.

Con estos dos criterios de preprocesamiento elegidos junto a la selección S4, se entrenaron las respectivas redes, para distinta cantidad de atributos de entrada. En la figura 7.1.7 se ve el resultado de las mejores configuraciones de selección y transformación, junto con el resultado de no aplicar ningún preprocesamiento. Ellos son:

- Los 65 atributos sin preprocesar.
- Los 30 primeros atributos de diferencia de valores medios de cada clase menos la suma de sus varianzas.
- Los 10 primeros de diferencia de valores medios de cada clase, para las primeras 30 componentes principales.
- La selección de 16 atributos obtenida con FSS usando naive Bayes.

Aquí se puede ver como el criterio elegido para el preprocesamiento de todo el conjunto de experimentos ha sido claramente el mejor. Resultó ser muy considerable la diferencia del error clasificación de preprocesar con FSS junto a naive Bayes frente a los demás criterios. De todos modos, ya sea la selección con S3 o proyectar las primeras 30 componentes principales y aplicar S2, obteniendo con ambos una eficiencia similar, es

superior a no realizar ningún preprocesamiento. Con lo cual, debido a lo que se ha dicho sobre la complejidad computacional, sumado a esta baja eficiencia, es totalmente necesario y útil aplicar algún tipo de reducción de la dimensión de la entrada.

7.2. Comparación del modelo propuesto con otros modelos de aprendizaje

Aquí se muestran experimentos donde se compara el modelo conexionista elegido para este trabajo con otras técnicas de aprendizaje: árboles de decisión (C4.5) y modelos estadísticos (naive Bayes). El mejor preprocesamiento para estos dos algoritmos, resultó ser también el mismo que para la red de Kohonen: la selección de atributos por intermedio de FSS junto con naive Bayes como clasificador.

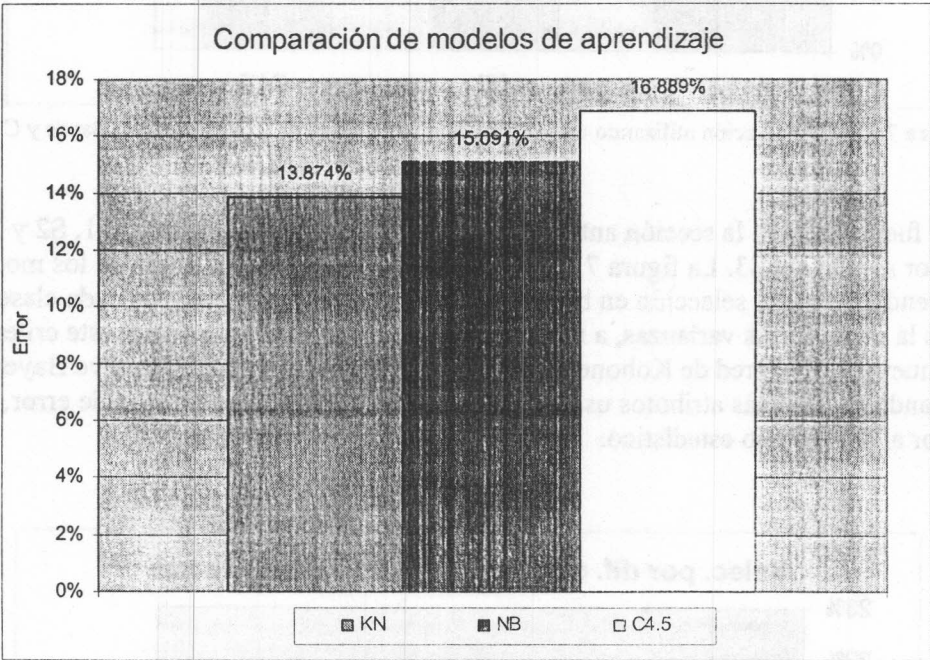


Figura 7.2.1. Comparación entre la red de Kohonen, naive Bayes y C4.5.

En la figura 7.2.1 se compara la mejor configuración de la red de Kohonen, contra naive Bayes y C4.5. El modelo conexionista empleado en este trabajo resultó ser el mejor de los tres, siguiéndolo luego naive Bayes.

En la figura 7.2.2 se hace la misma comparación pero utilizando la técnica de cross-validation de 10 hojas. Se muestra el promedio de error de clasificación y el mínimo obtenido de los respectivos 10 pares entrenamiento y evaluación. El resultado obtenido es similar al anterior, aunque la técnica empleada de cross-validation disminuye la diferencia entre los errores de los 3 modelos.

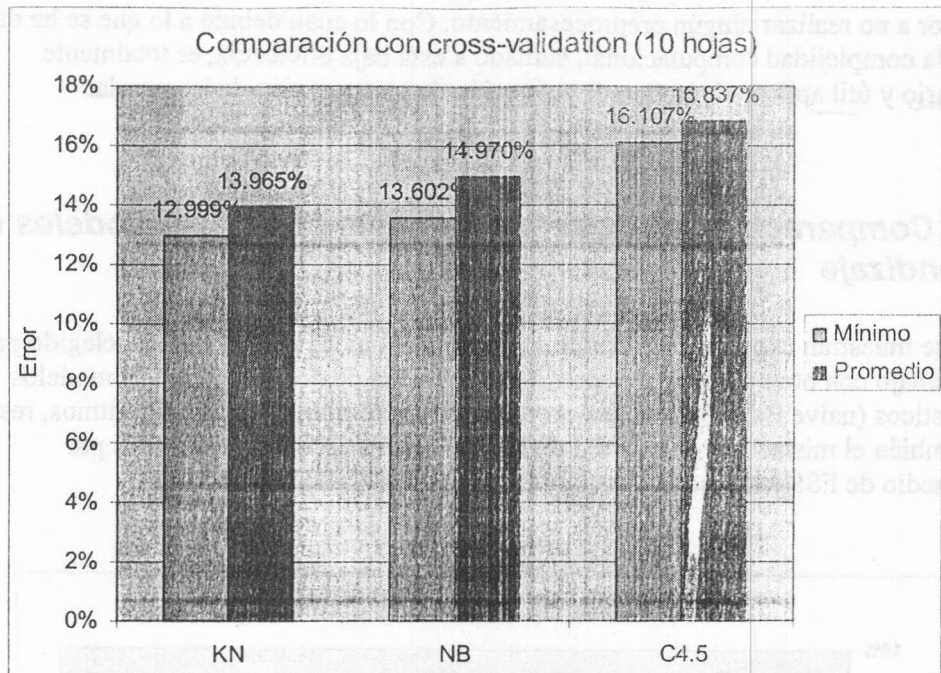


Figura 7.2.2. Comparación utilizando cross-validation entre la red de Kohonen, naïve Bayes y C4.5.

Como fuera dicho en la sección anterior, entre los 3 criterios de selección S1, S2 y S3, el mejor resultó ser S3. La figura 7.2.3 muestra cómo responde cada uno de los modelos de aprendizaje a esta selección en base a diferencia de valores medios de cada clase menos la suma de sus varianzas, a medida que se agregan atributos según este criterio. Aquí nuevamente la red de Kohonen resultó ser lo mejor. Mientras que naïve Bayes va mejorando cuanto más atributos usa, C4.5 logró inicialmente un pico bajo de error, inferior al del modelo estadístico.

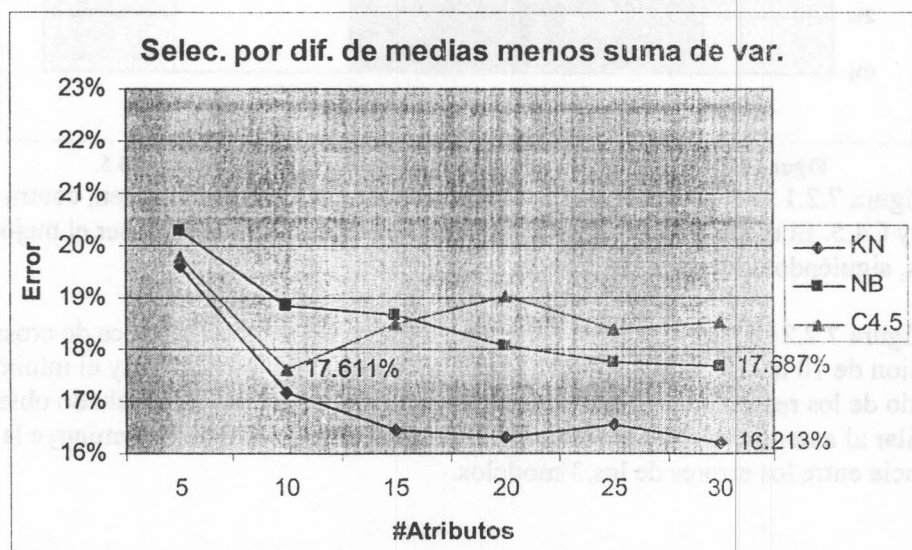


Figura 7.2.3. Evolución del error para los 3 modelos, utilizando el preprocesamiento de selección por diferencia de valores medios de cada clase menos suma de varianzas.

De la misma forma, se aplicó la transformación para proyectar y seleccionar las primeras 30 componentes principales, para luego utilizar la mejor selección, en este caso la diferencia de valores medios de cada clase. Al igual que en la figura anterior, la figura 7.2.4 muestra la comparación de las 3 técnicas de aprendizaje frente a esta transformación y selección de atributos.

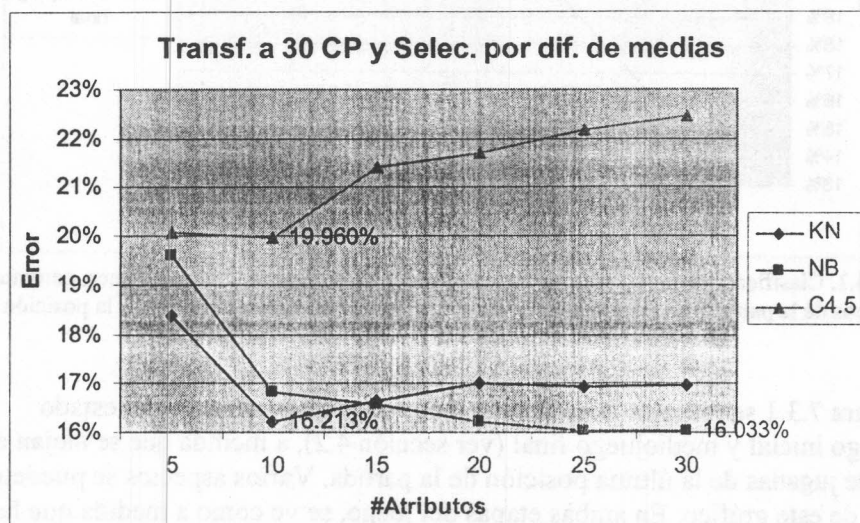


Figura 7.2.4. Evolución del error para los 3 modelos, utilizando el preprocesamiento de transformación a las 30 primeras componentes principales y luego selección por diferencia de valores medios de cada clase.

Para este preprocesamiento, naive Bayes resultó ser por una pequeña diferencia mejor a la red de Kohonen. En este como en los demás experimentos hechos para naive Bayes su evolución en la clasificación del error con la incorporación de atributos es siempre decreciente, cosa que no pasa con los demás modelos. En el caso de C4.5 crece notoriamente, con un error que dista mucho de lo alcanzado por Kohonen y naive Bayes.

¿por qué?

7.3. Experimentos con otras bases

Hasta aquí, todos los experimentos habían sido con la última posición de cada partida de la base mediojuego final (detallado en la sección 4.2). En el conjunto de experimentos siguiente, se aplica el mejor preprocesamiento y la mejor configuración de red obtenida a otros datos, es decir, a posiciones de otros momentos de la partida, y en relación a la cantidad de movidas que distan del final de la partidas (donde una movida está formada por la respectiva movida del blanco y del negro).

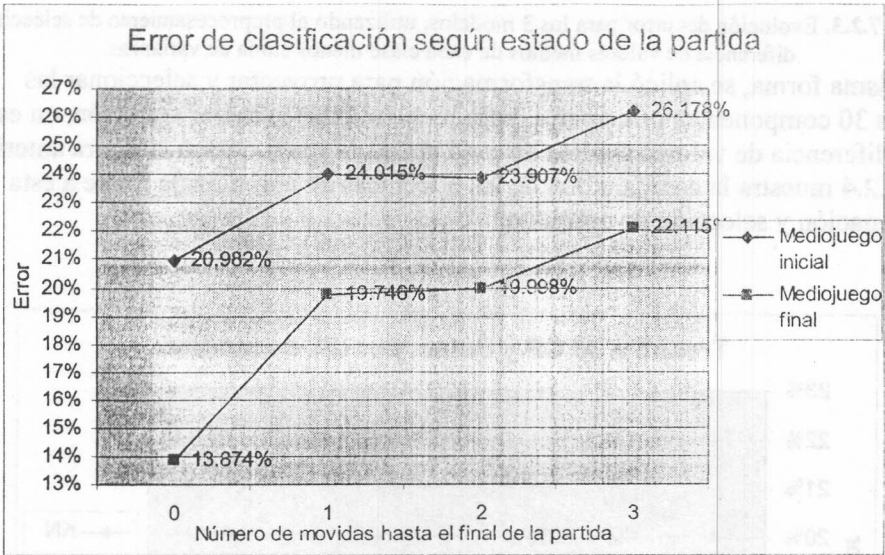


Figura 7.3.1. Clasificación de redes de Kohonen para distintos conjuntos de posiciones, perteneciendo a dos etapas de la partida, incrementando en cada conjunto el número de movidas a la posición final.

En la figura 7.3.1 se muestra la evolución del error para posiciones del estado mediojuego inicial y mediojuego final (ver sección 4.2), a medida que se alejan el número de jugadas de la última posición de la partida. Varios aspectos se pueden remarcar de este gráfico. En ambas etapas del juego, se ve como a medida que la posición se aleja de la final, el error crece de manera similar. En la etapa de mediojuego final se puede clasificar con bastante más precisión. Si bien este trabajo apunta a ayudar a construir una función de evaluación, se ve claramente que es una compleja tarea, ya que las posiciones que están a tan sólo 3 jugadas del final cuentan con un error bastante grande. Se puede decir que lo más dificultoso de toda esta labor, radica en el formular atributos posicionales adecuados.

*ideas para generar atributos para
todas las inst. del juego...*

8. Conclusiones

El problema tratado aquí es de mucha complejidad. En gran medida esto ocurre por el alto grado de solapamiento de las clases para el conjunto de datos utilizado. Es decir que ocurre frecuentemente que posiciones muy similares en los valores de los atributos pueden corresponder a partidas en las que gana tanto las blancas como las negras. Muestra de esto es que los experimentos fueron hechos con las últimas posiciones de cada partida, donde la ventaja es más clara para el ganador, ya que al ir alejándose unas pocas jugadas del final, el error crece considerablemente.

Con el modelo de aprendizaje propuesto, utilizando un mapa autoorganizado de Kohonen, se han obtenido los mejores resultados para los experimentos realizados, en comparación con otros dos modelos de aprendizaje, Naive Bayes y C4.5. Se probaron varios métodos de preprocesamiento de la entrada, resultando claramente superior la selección de atributos utilizando FSS, y Naive Bayes como su clasificador.

Trabajo futuro

Como se ha reiterado en varias oportunidades, lo más crítico del problema es una definición adecuada de los atributos que representan las posiciones. Por ello, queda como trabajo futuro el poder mejorar la definición de los atributos, ya sea modificando los que fueron utilizados para los experimentos, como el agregar nuevos.

Otra alternativa complementaria a los atributos *expertos*, es utilizar atributos obtenidos automáticamente, como por ejemplo la simple ubicación de las piezas en el tablero, o la agrupación de ellas formando nuevos atributos llamados comúnmente *chunks*, que son conjuntos de piezas ubicadas en casillas particulares que se presentan juntas con frecuencia en las posiciones (por Ej. Rg1, Tf1, f2, g2, h3 ó e4, d4, Cf3). Otra idea interesante para generar atributos automáticos es modelar la relación entre las distintas casillas del tablero, definiendo una función de distancia desde una casilla a cualquier otra, teniendo en cuenta a la vez qué pieza se encuentran en ella. Con ello se podría construir una red neuronal que aprenda una función de energía de las casillas.

! mi alg.

Donde también se notó una mejora considerable fue con el preprocesamiento de la entrada propuesto. Se podría experimentar con otros algoritmos de transformación y/o selección de los atributos.

Con las diferentes propuestas para que disminuya el error de clasificación, se espera que el aprendizaje y la evaluación puedan ser realizados con cualquier posición de la partida (no sólo con la última). Para ayudar a esto, se podría ponderar la clase en relación a la cantidad de jugadas hasta la última posición de la partida (utilizando la idea de diferencia temporal).

Para completar este trabajo, y que se pueda llegar a generar una función de evaluación, la clasificación de las posiciones tiene que dar un valor cuantificado de la ventaja para uno de los jugadores (es decir, que la clase sea continua), y el considerar las partidas terminadas en empate.

Si bien hay mucho por mejorar, dada la dificultad del problema, de esta forma se deja planteado un nuevo enfoque para obtener una función de evaluación, donde se busca aprender de manera automática a partir del conocimiento del hombre.

Agradecimientos

A Lili y mi familia por su apoyo; a Rocío por su gran colaboración, así como a otras personas que brindaron algún aporte; a Ariel por el despegue inicial; y a Enrique por tomar las riendas.

9. Bibliografía

- [1] M. Bain. "Learning Logical Exceptions in Chess". PhD thesis, Department of Statistics and Modelling Science, University of Strathclyde, Scotland, 1994.
- [2] J. Baxter, L. Weaver, A. Tridgell. "KnightCap: A chess program that learns by combining TD(lambda) with minimax search". Technical report, Department of Systems Engineering, Australian National University, Canberra, Australia, noviembre 1997.
- [3] H. J. Berliner, D. Kopec. "A taxonomy of concepts for evaluating chess strength". *Proceedings Supercomputing '90*: 336-343, Ed Northam.
- [4] A. Carrascal, D. Manrique, J. Ríos. "Neural Networks Evolutionary Learning in Chess Game". *MLMTA*, Carrascal, 148-153, 2003.
- [5] J. Fürnkranz. "Machine Learning in Computer Chess: The Next Generation". *International Computer Chess Association Journal* 19(3):147-161, septiembre 1996.
- [6] J. Fürnkranz. "Knowledge Discovery in Chess Databases: A Research Proposal". Technical Report OEFAI-TR-97-33, Austrian Research Institute for Artificial Intelligence, 1997.
- [7] B. Gaines. "Structured and Unstructured Induction with EDAGs". *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, Montreal, Canada (AAAI Press, San Mateo, CA), 1995.
- [8] S. Haykin. "Neural Networks: a Comprehensive Foundation, 2nd Ed". Upper Saddle River, NJ: Prentice Hall, 1999.
- [9] D. O. Hebb. "The Organization of Behavior". New York, Wiley, 1949.
- [10] Y. H. Kerner. "Learning Strategies for Explanation Patterns: Basic Game Patterns with Application to Chess". *ICCBR* 1995: 491-500.
- [11] R. Kohavi, G.H. John. "Wrappers for feature subset selection". *Artificial Intelligence* 97:1-2 (1997) 273-324.
- [12] T. Kohonen. "Self-Organized Formation of Topologically Correct Feature Maps". *Biological Cybernetics* 43, 59-69 (1982). Reimpreso en Anderson y Rosenfeld, 1988.
- [13] M. Kubat, J. Zizka. "Learning Middle Game Patterns in Chess: A Case Study". *IEA/AIE* 2000: 426-432.
- [14] S. Lazzeri, R. Heller. "A case-based approach for teaching high-level chess strategies". *IBERAMIA 96'*, Universidad de las Américas, Puebla, México, octubre 1996: 239-249.
- [15] R. Levinson. "Experience-Based Creativity". *Artificial Intelligence and Creativity: An Interdisciplinary Approach* (ed. T. Dartnall), pp. 161-179. Kluwer Academic Press, Boston, MA, 1994.
- [16] R. Levinson, R. Weber. "Chess neighborhoods, Function Combination, and reinforcement learning". In T. A. Marsland and I. Frank, editors, *Computers and Games: Proceedings of the 2nd International Conference (CG-01)*, volume 2063 of *Lecture Notes in Computer Science*, pages 133-150, Hamamatsu, Japan, 2001. Springer-Verlag.
- [17] T. Mitchell. "Machine Learning". McGraw-Hill, 1997.
- [18] E. Morales. "Learning Playing Strategies in Chess". *Computational Intelligence*, 12(1):65-87, 1996.

-
- [19] E. Robertsson. "Using Neural Networks in the Static Evaluation Function of a Computer Chess Program". Master's thesis, 2002.
 - [20] J. Ross Quinlan. "Induction of Decision Trees". *Machine Learning* 1(1): 81-106 (1986).
 - [21] J. Ross Quinlan. "C4.5: Programs for Machine Learning". Morgan Kaufmann, San Mateo, California, 1995.
 - [22] T. D. Sanger. "Optimal Unsupervised Learning in a Single-Layer Linear Feedforward Neural Network". *Neural Networks* 2, 459-473, 1989.
 - [23] A. D. Shapiro, Niblett. "Automatic Induction of Classification Rules for a Chess Endgame, In M.R.B.". Clarke (Ed.) *Advances in Computer Chess* 3, pp73-92 Oxford, Pergamon Press, 1982.
 - [24] A. D. Shapiro. "*Structured Induction in Expert Systems*". Wokingham, UK: Addison-Wesley, 1987.
 - [25] H. Simon, J. Schaeffer. "The Game of Chess". *Handbook of Game Theory with Economic Applications*, vol. 1, Robert J. Aumann and Sergiu Hart (editors), Elsevier Science Publishers, Netherlands, pp. 1-17, 1992.
 - [26] The Turing Institute. "Automated Acquisition of Concepts for the Description of Middle-Game Positions in Chess". Technical Report, Number TIRM-84-005, 1984.
 - [27] S. Thrun. "Learning to Play the Game of Chess". Universidad de Bonn, Alemania, 1995.
 - [28] D. E. Wilkins. "Using patterns and plans in chess". *Artificial Intelligence* 14 (1), 1980: 165-203.
 - [29] I. Witten, E. Frank. "Data Mining: practical machine learning tools and techniques with Java implementations". San Francisco, CA: Morgan Kaufmann, 1999.
 - [30] J. Zizka, M. Madr. "Learning Representative Patterns from Real Chess Positions: A Case Study". *IJCAI 2003*: 1374-1387.
 - [31] A. Zobrist, F. Carlson. "An advice-taking chess computer". *Scientific American*, 228(6):92-105, junio 1973.
 - [32] Chess Collect. <http://www.chesscollect.com/PGNDDownloads.html>
 - [33] Deep Blue, IBM Research, <http://www.research.ibm.com/deepblue/>.
 - [34] SCID. Shane's Chess Information Database. <http://scid.sourceforge.net/>.
 - [35] Valerie J. Easton and John H. McColl's Statistics Glossary v1.1 http://www.cas.lancs.ac.uk/glossary_v1.1/presdata.html#box.

Apéndice 1: Glosario ajedrecístico

Análisis táctico: Análisis de secuencias de jugadas precisas a partir de una posición, que puedan desencadenar en alguna buena jugada o combinación de jugadas.

Apertura, Mediojuego, Final: Estados generales en los que se suele dividir una partida de ajedrez.

Conceptos posicionales: nociones o criterios generales para evaluar las posiciones.

Desarrollar: Sacar las piezas de su casillero de origen.

ELO: Sistema de ranking internacional de ajedrez.

Enroque, Enrocar: Movimiento del ajedrez que se realiza entre el rey y una de las torres.

GM, MI, MF: Siglas utilizadas en ajedrez para designar los distintos títulos de categoría que puede tener un jugador: Gran Maestro, Maestro Internacional, Maestro FIDE (Federación Internacional de Ajedrez), respectivamente.

Material: Se denomina a las piezas disponibles en un momento del juego.

Movilidad: Cantidad de movidas disponibles.

Peón al paso: Movimiento del ajedrez en el cual un peón que está en la quinta fila, captura un peón del otro jugador ubicado en una columna contigua, que en la jugada previa avanzó dos casillas desde la fila 2 a la 4.

Piezas menores: denominadas así al caballo, alfil y torre.

PGN: del inglés portable game notation. Notación utilizada para describir partidas de ajedrez.

Apéndice 2: Consultas a investigadores

Varios de los autores de los trabajos aquí mencionados fueron consultados (vía correo electrónico) solicitándoles consejos o referencias en relación al tema aquí propuesto. Las siguientes son extractos de algunas respuestas:

Johannes Fürnkranz [5][6]: “Es una linda idea, pero los problemas más importantes que veo son: formular los atributos posicionales y armar un problema de clasificación (¿qué son las clases?).”

Michael Bain [1]: “Lo que propones suena ser un trabajo muy interesante. [...] podrías considerar un aprendizaje relacional (programación de lógica inductiva) en lugar de árboles de decisión (C4.5). Esto tiene el beneficio de permitir representaciones compactas para relaciones entre piezas.”

Andrew Tridgell [2]: “No he probado con este enfoque, pero imagino que la solución al problema se podría plantear con un conjunto robusto de atributos que puedan ser utilizados para distinguir posiciones de forma significativa. Si tuviera que probar esto, probablemente miraría atributos de control de casillas en lugar de atributos de ubicación de piezas, no obstante, sin pensarlo un poco, no se me ocurre ningún algoritmo obvio para comparar atributos.”

Eduardo Morales [18]: “[...] Esto me recuerda un poco el trabajo de A. Shapiro sobre ‘structured induction’ [24] en donde la idea era aprender a clasificar posiciones usando árboles

de decisión, sólo que los atributos podían ser en sí otros árboles aprendidos anteriormente. Lo usó solo para finales de juego [...]. Algo quizás más relacionado fue el trabajo posterior de S. Muggleton en su tesis con un sistema que se llamo DUCE.” “Lo mas difícil es definir los atributos. Si juegas bien ajedrez quizás no sea tan difícil. Para definirlos, es muy probable que tengas que usar una representación relacional que te sirva para extraer estas “propiedades” de posiciones (por ejemplo, la torres están alineadas, independientemente de su posición exacta en el tablero), pero para el algoritmo de aprendizaje, posiblemente con una representación atributo-valor sea suficiente (torres-alineadas: con valores de verdadero o falso).”

Santos Gerardo Lázzari Menéndez [14]: “Me parece muy interesante tu tema de tesis [...]. Respecto al uso de aprendizaje, no lo intenté en mi trabajo, principalmente debido a que decidí enfocarme en el medio juego y consideré que la explosión combinatoria en esta fase del juego haría muy difícil el obtener resultados razonables con un algoritmo de aprendizaje. Sin embargo, si piensas enfocarte en finales, entonces creo que hay mejores posibilidades de éxito con estos enfoques, tal como en el trabajo de Eduardo Morales o en ID3.”

“Uno de los problemas que menciona Kubat/Zizka es que hay muy pocas posiciones en que se encuentre el patrón sobre el que estaban trabajando [...]. Ellos usaron bases de datos de partidas de programas comerciales y atribuían el problema a que era un patrón muy conocido y jugadores de torneo usualmente no lo permiten. Creo que si te limitas a partidas de GMs, este tipo de problema se te puede magnificar.”

Miroslav Kubat [13]: “El aspecto más dificultoso de tu trabajo es cómo describir las posiciones. La intuición humana no es muy buena en esto y, más aún, algunos atributos que tienen sentido para los humanos (por Ej., "predominancia en el centro" o "un caballo más activo") son sorprendentemente difíciles de implementar en un programa de computación. Por lo tanto, lo que se me ocurre es que uno debe primero tratar de inducir algunos conceptos útiles como los anteriores. O tal vez la máquina podría hacerlo mejor con diferentes atributos a aquellos con los que los humanos cuentan.”

“Tengo la siguiente idea: los jugadores humanos perciben la posición en el tablero en términos de *chunks* –típicas configuraciones de piezas son "Tf1, Rg2, f2, g2, h2" o "Cc3, d4, e4". Un maestro o gran maestro sabe miles de *chunks* diferentes adquiridos durante los años de práctica y estudio. Cuanto mejor sean los *chunks*, mejor será el jugador –mucho de esto lo sabemos de la psicología del ajedrez. Por lo tanto parece ser que necesitamos algún mecanismo que emule la "adquisición de *chunks*" del jugador de ajedrez.”

Jan Zizka [13][30]: “[...] Tengo una idea de entrenar árboles de decisión y luego convertirlos en una red neuronal la cual (tal vez) pueda incrementar la habilidad de reconocer sacrificios de piezas ganadores/perdedores, etc. [...] Quizás, el clasificador (naive) Bayes también pueda proveer buenos resultados. Pero todo lo que he mencionado significa mucho trabajo, demasiado para tan sólo una tesis de graduación.”

“De cualquier modo, los árboles de decisión parecen ser un buen punto de partida. No obstante, también sería bueno experimentar con otros métodos. Sugeriría probar un clasificador (naive) Bayes y/o métodos IBL. Todo depende de cómo seleccionas los atributos importantes (significativos) para tu tipo de posiciones. Los árboles de decisión pueden seleccionar atributos, sin embargo, entre los atributos seleccionados automáticamente suelen haber atributos nada significativos [...]. Para los árboles de decisión, éste es un problema ya que este tipo de atributo fácilmente produce una clase homogénea, aún si ese atributo no es importante.”

“[...] Usamos el programa Fritz 7 para analizar posiciones individuales (¡un montón de tiempo!) y si el programa encontraba que Axh7+ era la mejor movida, la usamos como un ejemplo positivo, sino fuera así, entonces negativo. [...]. Una cosa importante es asegurarse de que los ejemplos positivos son realmente partidas ganadas, y los negativos son realmente partidas perdidas. (sólo una experiencia limitada a nuestras posiciones con Axh7+, ¿tal vez sea/no sea una regla general?)”

Jonathan Schaeffer [25]: “El problema que estás investigando es complicado y probablemente no sea exitoso. *Chunking* es un tema importante, pero soy escéptico en que las redes neuronales puedan hacer un buen trabajo y, más aún, que esto pueda ser utilizado para predecir el resultado de un partido, que no sean posiciones triviales. No quiero sonar desalentador, pero tu objetivo es muy difícil y probablemente será complicado de evaluar.”

Se pudo recolectar unas cuantas opiniones de respetados investigadores del área en cuestión. Si bien sus respuestas fueron diversas, todos coincidieron en que el tema abordado aquí es sumamente complejo.

Apéndice 3: Definición de atributos

Atributos

Los siguientes son los 65 atributos utilizados para todos los experimentos, detallados más adelante.

Nº	Atributo	Nº	Atributo
A01	pawns_nbr_diff	A34	rook_connected_diff
A02	knight_nbr_diff	A35	p_queenSideMajority
A03	light_bishop_nbr_diff	A36	p_kingSideMajority
A04	dark_bishop_nbr_diff	A37	p_islands_diff
A05	rook_nbr_diff	A38	p_solatedClosed_diff
A06	queen_nbr_diff	A39	p_solatedSOpen_diff
A07	move_time_diff	A40	p_doubleNolso_diff
A08	mobility_diff	A41	p_advanced_diff
A09	development_diff	A42	p_hanging_diff
A10	space_ctrl_diff	A43	p_passedBlocked_diff
A11	space_advantage_diff	A44	p_passedNotBlocked_diff
A12	center_occup_diff	A45	ctrl_mincontrol_diff
A13	center_ext_occup_diff	A46	ctrl_advantage_diff
A14	centrality_knight_diff	A47	ctrl_absolute_diff
A15	centrality_bishop_diff	A48	ctrl_queenSideAttackMinControlDiff
A16	centrality_rook_diff	A49	ctrl_queenSideAttackAdvantageDiff
A17	centrality_queen_diff	A50	ctrl_queenSideAttackAbsoluteDiff
A18	centrality_king_w	A51	queenSideAttackMobilityDiff
A19	centrality_king_b	A52	ctrl_kingSideAttackMinControlDiff
A20	tropismKnight_diff	A53	ctrl_kingSideAttackAdvantageDiff
A21	tropismRook_diff	A54	ctrl_kingSideAttackAbsoluteDiff
A22	tropismQueen_diff	A55	kingSideAttackMobilityDiff
A23	kingDefenseKnight_diff	A56	ctrl_centerSectorAttackMinControlDiff
A24	kingDefenseBishop_diff	A57	ctrl_centerSectorSideAttackAdvantageDiff
A25	kingDefenseQueen_diff	A58	ctrl_centerSectorSideAttackAbsoluteDiff
A26	kingSquaresCtrl_diff	A59	centerSectorAttackMobilityDiff
A27	kingPawns_w	A60	mobil_pawn_diff
A28	kingPawns_b	A61	mobil_queen_diff
A29	outpost_knight_diff	A62	mobil_rook_diff
A30	outpost_bishop_diff	A63	mobil_knight_diff
A31	rook7Rank_diff	A64	mobil_bishop_diff
A32	rookOpen_diff	A65	mobil_king_diff
A33	rookSOpen_diff		

El sufijo “diff” indica que se expresa la diferencia entre el valor del atributo para las blancas menos el respectivo para las negras.

columna: 1, 2, 3, 4, 5, 6, 7, 8.

Diferencia en sectores de ataque (este concepto es utilizado por algunos atributos). Para obtener este valor, se restan los valores del atributo en cuestión en cada uno de los sectores correspondientes, por el blanco en primer lugar, y luego el negro:

Ataque en sector dama	= FlancoDamaBlanco - FlancoDamaNegro
Ataque en sector rey	= FlancoReyBlanco - FlancoReyNegro
Ataque en sector centro	= CentroBlanco - CentroNegro
Ataque en centro principal	= CentroPrincipal - CentroPrincipal

.....

Atributos de Peones

Variables usadas para las expresiones

p: peón
C: color(p)
CO: color_opuesto(C)
f: fila(p, C)
c: columna(p)

mayoría de flanco (sideMajority): relación de cantidad de peones entre los bandos, para cada flanco (dama y rey). Hay 3 valores posibles:

- *mayoría blanca (WhiteMajority):* mayor cantidad de peones blancos en el flanco.
- *mayoría negra (BlackMajority):* mayor cantidad de peones negros en el flanco.
- *no hay mayoría (NoMajority):* igual cantidad de peones en el flanco.

flanco dama: col1 = 1; col2 = 4
flanco rey: col1 = 5; col2 = 8

sideMajority: WhiteMajority
cantidad_peones(col1, col2, Blanco) > cantidad_peones(col1, col2, Negro)

sideMajority: BlackMajority
cantidad_peones(col1, col2, Blanco) < cantidad_peones(col1, col2, Negro)

sideMajority: NoMajority
cantidad_peones(col1, col2, Blanco) = cantidad_peones(col1, col2, Negro)

islas (islands): islas de peones, grupos de peones que a los de los extremos los separa una o más columnas del resto.

Todo peón p tal que
Existe columnas c1 y c2 tal que
{ columna(p) >= c1 y columna(p) <= c2 y
(para toda columna c / c1 <= c y c <= c2, cantidad_peones(c, C) > 0) y
c1 = 1 o cantidad_péon(c1-1, C) = 0 y
c2 = 8 o cantidad_péon(c1+1, C) = 0 }

avanzado (advanced): peón avanzado, supera la cuarta fila.

Fila(p, C) > 4

p.iii) Para los siguientes atributos se usarán los predicados:
semiabierta(c, C), cerrada(c), aislado(p).

Aislado en Semiabierta (singleIsoSOpen): peón aislado en columna semiabierta.

Aislado en Cerrada (singleIsoClosed): peón aislado en columna cerrada.

p: singleIso

cantidad_peones(c, C) = 1 y aislado(p)

Doblados no aislados (doubleNoIso): peones doblados no aislados.

p: double

cantidad_peones(c, C) = 2 y NO aislado(p)

Retrasado en Semiabierta (backwardSOpen): Peón retrasado en columna semiabierta. Se toma como peón aislado.

Retrasado en Cerrada (backwardClosed): Peón retrasado en columna cerrada. Se toma como peón aislado.

p: backward

Existe peón pco / (color(pco) = CO) y

(c > 1 y columna(pco) = c-1) o (c < 8 y columna(pco) = c+1) y

(fila(pco, C) > f+1 y fila(pco, C) < f+4) y

{ No Existe peón p1 / columna(p1) = p y (fila(p1, C) > f y fila(p1, C) < fila(pco, C))

} y

{ No Existe peón pc / (color(pc) = C) y

[(c > 1 y columna(pc) = c-1) o (c < 8 y columna(pc) = c+1)] y

[fila(pco)-1 > fila(pc, C) y fila(pc, C) >= fila(p)-1] } y

{ Si Existe peón pc / (color(pc) = C) y

[(c > 1 y columna(pc) = c-1) o (c < 8 y columna(pc) = c+1)] y (fila(p) > fila(pc,

C)+1)

Entonces Existe pco2 / (color(pco2) = CO) y

(columna(pco2) = columna(pc) y fila(pco2) = fila(p) - 1) }

{ p = backwardSOpen y

No Existe peón pco / color(pco) = CO y columna(pco) = c y fila(pco, C) > f } o

{ p = backwardClosed y

Existe peón pco / color(pco) = CO y columna(pco) = c y fila(pco, C) > f }

pasado bloqueado (passedBlocked): peón pasado bloqueado
pasado no bloqueado (passedNotBlocked): peón pasado no bloqueado

p: passed

No Existe peón pco / color(pco) = CO y
 [columna(pco)=c y fila(pco, C) > f] o
 [((c>1 y columna(pco)=c-1) o (c<8 y columna(pco)=c+1)) y fila(pco, C) > f+1]
 y
 { Si Existe peón pco / color(pco) = CO y
 [((c>1 y columna(pco)=c-1) o (c<8 y columna(pco)=c+1)) y fila(pco, C) = f+1]
 entonces Existe peón pc / color(pc) = C y
 [((c>1 y columna(pco)=c-1) o (c<8 y columna(pco)=c+1)) y (fila(pco, C) = f-1)
] } y
 { p = passedBlocked y
 Existe pieza P / color(P) = CO y columna(P)=c y fila(P, C) = f+1 } o
 { p = passedNotBlocked y
 No Existe pieza P / color(P) = CO y columna(P)=c y fila(P, C) = f+1 }

colgantes (hanging): peones colgantes.

Existe peón p1 y p2 / (color(p1) = C y color(p2) = C y fila(p1) = 4 y fila(p2) = 4 y
 columna(p1) = columna(p2)+1 y columna(p1) > 1 y columna(p2) < 8) y
 (No Existe peón p3 / color(p3) = C y
 (columna(p3) = columna(p1) - 1 o columna(p3) = columna(p2) + 1)

Atributos de Material

peones (pawns_nbr): cantidad de peones.
caballos (knight_nbr): cantidad de caballos.
alfil de casillas blancas (light_bishop_nbr): cantidad de alfiles de casillas blancas.
alfil de casillas negras (dark_bishop_nbr): cantidad de alfiles de casillas negras.
torres (rook_nbr): cantidad de torres.
damas (queen_nbr): cantidad de damas.

Atributos de Movimiento

tiempo (moveTime): suma total de movimientos que deben realizar
 independientemente todas las piezas desde su casilla de origen hasta su casilla actual.

desarrollo (development): cantidad de figuras que están fuera de su casilla de inicio
 (exceptuando al rey).

movilidad (mobility): suma total de movimientos disponibles.

movilidad de pieza (pieceMobility): cantidad de movimientos disponibles por cada tipo de pieza.

movilidad por sectores (sectorAttackMobilityDiff): diferencia de movilidad en cada uno de los 4 sectores descriptos en la “definición de predicados”. Se toma en cuenta la movilidad de todas las piezas, exceptuando rey y peones.

.....

Atributos de Espacio

control de espacio (spaceControl): suma total de casillas a las que tienen alcance todas las piezas (de un bando). Equivalente a la movilidad de todas las figuras, considerando también las casillas de captura de los peones, y las casillas alcanzadas por cualquier pieza que están ocupadas por otra pieza propia.

ventaja de espacio (spaceAdvantage): suma total de casillas en cada columna desde la fila 2 hasta la casilla anterior a un propio peón, o si hay un peón rival, hasta esa casilla incluida, o sino hay peones hasta la fila 7.

.....

Atributos de Control

control absoluto (absolute): Cantidad de casillas con dominio absoluto. Un jugador tiene control absoluto de una casilla si al menos la controla con alguna pieza, y el oponente con ninguna.

ventaja de control (advantage): Cantidad de casillas con ventaja de control. Un jugador tiene ventaja en el control de una casilla si tiene control absoluto, o si la menor pieza con la cual la controla es menor que la menor pieza con la que controla el oponente.

control mínimo (minControl): Cantidad de casillas con control mínimo. Un jugador tiene control mínimo de una casilla si al menos una pieza controla la casilla.

Diferencias de control en sectores de ataque: diferencia de control en cada uno de los 4 sectores descriptos en la “definición de predicados”, para cada uno de los tres controles anteriores: absoluto (sectorAttackAbsoluteDiff), ventaja (sectorAttackAdvantageDiff), y mínimo (sectorAttackMinControlDiff)

.....

Atributos de Centro

centro: filas 4-5; columnas 4-5
centro extendido: filas 3-6; columnas 3-6

ocupación del centro (centerOccupation): cantidad de piezas ubicadas en las casillas del centro.

ocupación del centro extendido (extCenterOccupation): cantidad de piezas ubicadas en las casillas del centro extendido.

centralización de torres (centralityRook): valor en base a la centralización de columna de ambas torres. El valor de cada torre está dado por su ubicación en el tablero, correspondiente con la siguiente asignación para cada columna:

1	2	3	4	5	6	7	8
1	2	3	4	4	3	2	1

centralización de alfiles (centralityBishop): valor en base a la centralización de ambos alfiles. El valor de cada alfil está dado por su ubicación en el tablero, correspondiente con la siguiente asignación para cada casilla:

	1	2	3	4	5	6	7	8
1	3	2	1	0	0	1	2	3
2	2	4	5	5	5	5	4	2
3	1	5	8	10	10	8	5	1
4	0	5	10	11	11	10	5	0
5	0	5	10	11	11	10	5	0
6	1	5	8	10	10	8	5	1
7	2	4	5	5	5	5	4	2
8	3	2	1	0	0	1	2	3

centralización de caballos (centralityKnight): valor en base a la centralización de ambos caballos.

centralización de la dama (centralityQueen): valor en base a la centralización de la dama.

centralización del rey (centralityKing): valor en base a la centralización del rey.

El valor de cada caballo/dama/rey está dado por su ubicación en el tablero, correspondiente con la siguiente asignación para cada casilla:

	1	2	3	4	5	6	7	8
1	0	1	2	3	3	2	1	0
2	1	3	4	5	5	4	3	1
3	2	4	6	7	7	6	4	2
4	3	5	7	9	9	7	5	3
5	3	5	7	9	9	7	5	3
6	2	4	6	7	7	6	4	2
7	1	3	4	5	5	4	3	1
8	0	1	2	3	3	2	1	0

Atributos de Rey

control casillas del rey (kingSquaresCtrl): cantidad de veces que estan controladas las casillas adyacentes al rey adversario por alguna pieza propia. Se considera adyacente toda casilla a la que esta a un movimiento de rey de distancia.

peones del rey (kingPawns): a mayor valor, menor defensa de peones cercanos al rey. Si la dama opuesta no esta en juego, o el rey esta en la columna dama o rey, el valor es cero. Caso contrario, se toma en cuenta la ubicacion de los peones alfil, caballo y torre del flanco en que se encuentre el rey. El valor de este atributo es la suma total de la cantidad de casillas avanzadas por cada uno de estos peones (si son 3 o mas, o no esta el peon, se considera 3).

ataque al rey de caballo (tropismKnight): cercana de caballos al rey adversario. El valor se obtiene sumando la distancia en filas y en columnas de las casillas de cada caballo al rey opuesto.

CR: Caballo Rey
CD: Caballo Dama
RA: Rey Adversario

$$\text{tropismKnight} = 15 - (\text{abs}(\text{fila}(\text{CR}, \text{C}) - \text{fila}(\text{RA}, \text{C})) + \text{abs}(\text{columna}(\text{CR}) - \text{columna}(\text{RA}))) + 15 - (\text{abs}(\text{fila}(\text{CD}, \text{C}) - \text{fila}(\text{RA}, \text{C})) + \text{abs}(\text{columna}(\text{CD}) - \text{columna}(\text{RA})))$$

ataque al rey de dama (tropismQueen): cercana de dama al rey adversario. El valor se obtiene la mınima distancia en filas y en columnas de las casillas de la dama al rey opuesto.

D: Dama
RA: Rey Adversario

$$\text{tropismQueen} = 8 - \min(\text{abs}(\text{fila}(\text{D}, \text{C}) - \text{fila}(\text{D}, \text{C})) + \text{abs}(\text{columna}(\text{D}) - \text{columna}(\text{RA})))$$

ataque al rey de torre (tropismRook): cercanía de torres al rey adversario. El valor se obtiene sumando la distancia en columnas de las casillas de cada torre al rey opuesto, si están en una columna abierta o semiabierta.

TR: Torre Rey
TD: Torre Dama
RA: Rey Adversario

$$\text{tropismRook} = 8 - \text{abs}(\text{columna}(\text{TR}) - \text{columna}(\text{RA})) + 8 - \text{abs}(\text{columna}(\text{TD}) - \text{columna}(\text{RA}))$$

defensa del rey de caballo (kDefenseKnight): cercanía de caballos al rey propio. El valor se obtiene sumando la distancia en filas y en columnas de las casillas de cada caballo al rey propio.

CR: Caballo Rey
CD: Caballo Dama
R: Rey

$$\begin{aligned} \text{kDefenseKnight} = & 15 - (\text{abs}(\text{fila}(\text{CR}, \text{C}) - \text{fila}(\text{R}, \text{C})) + \text{abs}(\text{columna}(\text{CR}) - \text{columna}(\text{R}))) + \\ & 15 - (\text{abs}(\text{fila}(\text{CD}, \text{C}) - \text{fila}(\text{R}, \text{C})) + \text{abs}(\text{columna}(\text{CD}) - \text{columna}(\text{R}))) \end{aligned}$$

defensa del rey de (kDefenseBishop): cercanía de alfiles al rey propio. El valor se obtiene sumando la distancia en filas y en columnas de las casillas de cada alfil al rey propio.

AR: Alfil Rey
AD: Alfil Dama
R: Rey

$$\begin{aligned} \text{kDefenseBishop} = & 15 - (\text{abs}(\text{fila}(\text{AR}, \text{C}) - \text{fila}(\text{R}, \text{C})) + \text{abs}(\text{columna}(\text{AR}) - \text{columna}(\text{R}))) + \\ & 15 - (\text{abs}(\text{fila}(\text{AD}, \text{C}) - \text{fila}(\text{R}, \text{C})) + \text{abs}(\text{columna}(\text{AD}) - \text{columna}(\text{R}))) \end{aligned}$$

defensa del rey de dama (kDefenseQueen): cercanía de la dama al rey propio. El valor se obtiene de la mínima distancia en columnas de las casillas de la dama al rey propio.

D: Dama
R: Rey

$$\text{tropismQueen} = 8 - \text{abs}(\text{columna}(\text{D}) - \text{columna}(\text{R}))$$

Atributos Especiales

torre en septima fila (rook7Rank): cantidad de torres/dama en septima fila.

torre en abierta (rookOpen): cantidad de torres en columnas abiertas. Si ambas torres están en la misma columna se cuentan 3 (en vez de 2).

torre en semiabierta (rookSOpen): cantidad de torres en columnas semiabiertas. Si ambas torres están en la misma columna se cuentan 3 (en vez de 2).

torres conectadas (rookConnected): si las torres están conectadas (se defienden mutuamente), vale 1, sino 0.

Alfil (outpostBishop) y caballo (outpostKnight) en casilla fuerte: una pieza menor (caballo, alfil) que no puede ser echada por un peón enemigo, y que está defendida por un peón propio. Es más fuerte cuando es defendido por dos peones o si el contrincante no tiene una pieza menor para atacarla.

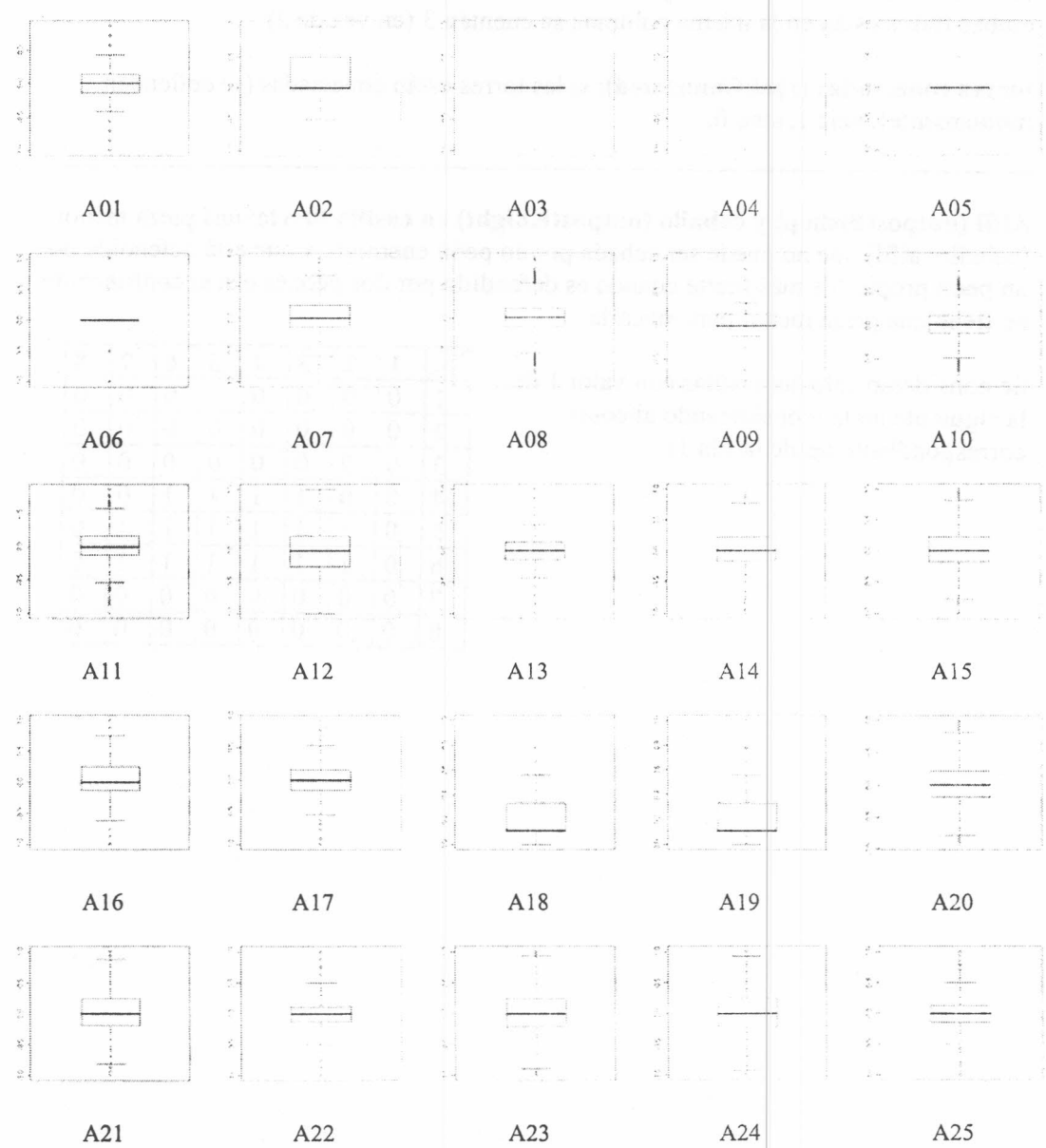
Se consideran sólo las casillas con valor 1 de la siguiente tabla (considerando al color correspondiente desde la fila 1):

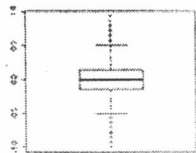
	1	2	3	4	5	6	7	8
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	1	1	1	1	0	0
5	0	1	1	1	1	1	1	0
6	0	1	1	1	1	1	1	0
7	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0

Apéndice 4: Estadísticas de los datos utilizados

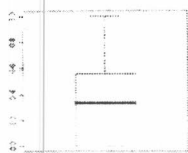
Boxplot

A continuación se muestra una serie de gráficos boxplot para cada uno de los 65 atributos definidos, para la base de las 31.550 posiciones utilizadas en los experimentos. Las respectivas referencias para cada atributo están detalladas en el apéndice 3.

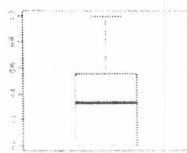




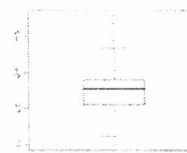
A26



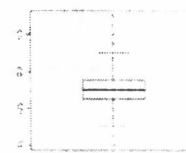
A27



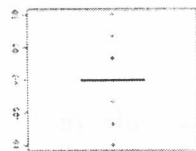
A28



A29



A30



A31



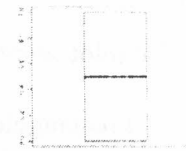
A32



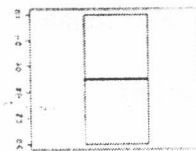
A33



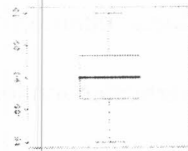
A34



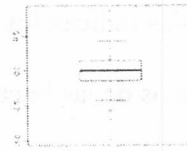
A35



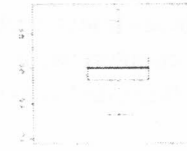
A36



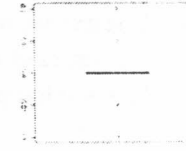
A37



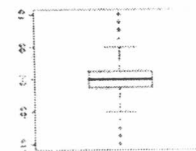
A38



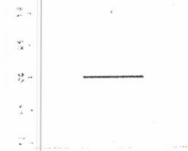
A39



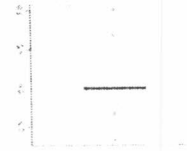
A40



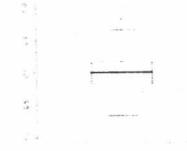
A41



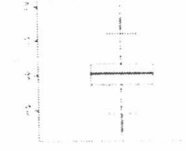
A42



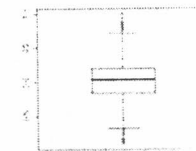
A43



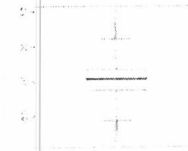
A44



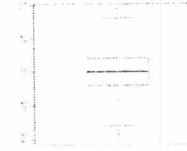
A45



A46



A47



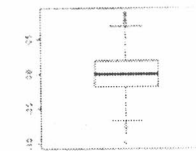
A48



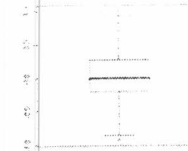
A49



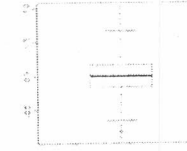
A50



A51



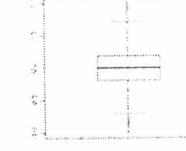
A52



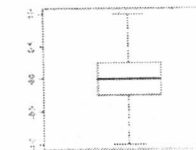
A53



A54



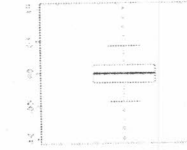
A55



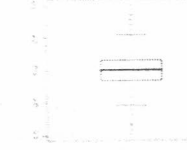
A56



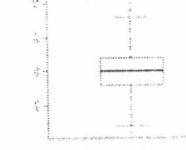
A57



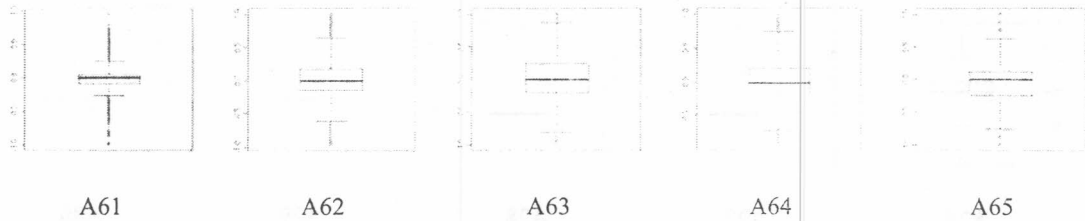
A58



A59



A60



El boxplot se interpreta de la siguiente manera:

- El rectángulo contiene el 50% central de los datos. Los bordes superior e inferior marcan sendos cuartiles superior e inferior, respectivamente. El rango entre medio es conocido como el rango intercuartil.
- La línea en el rectángulo indica el valor de la mediana.
- Los extremos de las líneas verticales indican los valores mínimo y máximo, siendo 1.5 veces el rango intercuartil.
- Los puntos por fuera de los extremos de las líneas verticales son los *outliers*.

Matriz de correlación

En la matriz siguiente se muestran las correlaciones de los 65 atributos y la clase tomados de a pares, para el mismo conjunto de datos utilizados para generar los *boxplots*.

	A01	A02	A03	A04	A05	A06	A07	A08	A09	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19	A20	A21	A22	A23	A24	A25	A26	A27	A28	A29	A30	A31	A32	A33
A01	1.00	-0.08	-0.03	-0.03	-0.08	0.02	0.37	0.12	-0.08	0.27	-0.32	0.21	0.34	-0.03	0.03	-0.03	0.09	-0.06	0.01	-0.03	-0.17	0.08	-0.10	-0.06	-0.07	0.22	-0.13	0.16	0.18	0.17	0.11	0.05	-0.32
A02	-0.08	1.00	-0.30	-0.31	-0.21	-0.11	0.22	0.00	0.27	0.03	-0.03	0.05	0.11	0.87	-0.33	-0.15	-0.02	0.07	-0.03	0.93	-0.12	-0.11	0.95	-0.42	-0.04	-0.05	0.01	-0.04	0.19	0.10	-0.06	-0.11	-0.01
A03	-0.03	-0.30	1.00	-0.15	-0.16	-0.07	0.06	0.10	0.15	0.12	0.07	0.02	0.04	-0.26	0.47	-0.13	-0.03	0.01	0.00	-0.28	-0.06	-0.06	-0.28	0.61	-0.04	0.03	0.05	-0.02	-0.13	-0.12	0.00	-0.06	-0.02
A04	-0.03	-0.31	-0.15	1.00	-0.17	-0.08	0.06	0.09	0.14	0.11	0.06	0.04	0.06	-0.27	0.43	-0.13	-0.02	0.02	0.02	-0.28	-0.07	-0.06	-0.29	0.61	-0.05	0.05	0.00	-0.07	0.04	0.06	-0.01	-0.07	-0.02
A05	-0.08	-0.21	-0.16	-0.17	1.00	-0.35	-0.10	0.12	0.13	0.15	0.01	-0.06	-0.10	-0.19	-0.20	0.71	-0.18	-0.01	0.03	-0.20	0.45	-0.25	-0.21	-0.24	-0.23	-0.01	0.09	-0.08	-0.13	-0.11	0.09	0.38	0.17
A06	0.02	-0.11	-0.07	-0.08	-0.35	1.00	-0.04	0.15	-0.13	0.14	-0.02	-0.01	-0.01	-0.09	-0.08	-0.25	0.40	-0.03	-0.01	-0.09	-0.15	0.67	-0.11	-0.11	0.57	0.05	-0.19	0.21	0.00	0.00	0.07	-0.13	-0.05
A07	0.37	0.22	0.06	0.06	-0.10	-0.04	1.00	0.24	0.38	0.37	0.40	0.26	0.44	0.23	0.15	-0.01	0.07	0.06	-0.05	0.27	-0.04	0.01	0.18	0.08	-0.03	0.18	0.07	-0.08	0.18	0.12	0.05	0.05	-0.16
A08	0.12	0.00	0.10	0.09	0.12	0.15	0.24	1.00	0.42	0.92	-0.08	0.18	0.28	0.12	0.27	0.15	0.31	0.07	-0.08	0.06	0.23	0.13	-0.04	0.10	-0.05	0.31	-0.07	0.06	0.15	0.13	0.22	0.34	-0.08
A09	-0.08	0.27	0.15	0.14	0.13	-0.13	0.38	0.42	1.00	0.46	0.01	0.09	0.16	0.28	0.28	0.30	0.00	0.01	0.07	0.30	0.23	-0.07	0.23	0.17	-0.13	0.21	0.03	-0.09	0.07	0.08	0.09	0.17	0.06
A10	0.27	0.03	0.12	0.11	0.15	0.14	0.37	0.92	0.46	1.00	-0.01	0.27	0.44	0.16	0.31	0.17	0.38	0.12	-0.10	0.08	0.16	0.08	0.01	0.14	0.02	0.26	-0.06	0.06	0.16	0.14	0.17	0.31	-0.14
A11	-0.32	-0.03	0.07	0.06	0.01	-0.02	0.40	-0.08	0.01	-0.01	1.00	0.15	0.18	-0.02	0.05	-0.02	0.03	0.10	-0.10	-0.05	-0.02	-0.09	-0.01	0.13	0.10	-0.17	0.22	-0.16	-0.07	-0.12	-0.14	-0.07	0.06
A12	0.21	0.05	0.02	0.04	-0.06	-0.01	0.26	0.18	0.09	0.27	0.15	1.00	0.57	0.18	0.19	0.01	0.24	0.05	-0.05	0.08	-0.05	-0.03	0.04	0.03	0.02	0.04	0.02	0.01	0.14	0.13	-0.04	0.02	-0.13
A13	0.34	0.11	0.04	0.06	-0.10	-0.01	0.44	0.28	0.16	0.44	0.18	0.57	1.00	0.24	0.30	0.00	0.33	0.15	-0.14	0.12	-0.09	-0.05	0.10	0.08	0.06	0.02	0.02	-0.02	0.18	0.16	-0.10	0.03	-0.14
A14	-0.03	0.87	-0.26	-0.27	-0.19	-0.09	0.23	0.12	0.28	0.16	-0.02	0.18	0.24	1.00	-0.27	-0.13	-0.01	0.07	-0.03	0.86	-0.09	-0.08	0.81	-0.37	-0.05	0.01	0.00	-0.02	0.37	0.10	-0.02	-0.07	-0.03
A15	0.03	-0.33	0.47	0.48	-0.20	-0.08	0.15	0.27	0.28	0.31	0.05	0.19	0.30	-0.27	1.00	-0.15	-0.02	0.04	0.00	-0.29	-0.07	-0.05	-0.31	0.65	-0.06	0.13	0.01	-0.04	-0.02	0.24	0.02	-0.05	-0.04
A16	-0.03	-0.15	-0.13	-0.13	0.71	-0.25	-0.01	0.15	0.30	0.17	-0.02	0.01	0.00	-0.13	-0.15	1.00	-0.09	-0.04	0.06	-0.12	0.51	-0.17	-0.15	-0.20	-0.18	0.04	0.05	-0.07	-0.08	-0.06	0.06	0.43	0.11
A17	0.09	-0.02	-0.03	-0.02	-0.16	0.40	0.07	0.31	0.00	0.36	0.03	0.24	0.33	-0.01	-0.02	-0.09	1.00	0.04	-0.08	-0.01	-0.08	0.23	-0.03	-0.03	0.36	0.00	-0.08	0.10	0.03	0.01	-0.06	-0.02	-0.07
A18	-0.06	0.07	0.01	0.02	-0.01	-0.03	0.06	0.07	0.01	0.12	0.10	0.05	0.15	0.07	0.04	-0.04	0.04	1.00	0.14	0.01	-0.09	-0.11	0.12	0.08	0.11	-0.31	-0.18	-0.14	0.01	-0.01	-0.02	-0.03	0.02
A19	0.01	-0.03	0.00	0.02	0.03	-0.01	-0.05	-0.06	0.07	-0.10	-0.10	-0.05	-0.14	-0.03	0.03	0.06	-0.06	0.14	1.00	0.04	0.13	0.07	-0.07	-0.02	-0.14	0.33	-0.15	-0.22	-0.01	-0.01	0.01	0.04	0.02
A20	-0.03	0.93	-0.28	-0.28	-0.20	-0.09	0.27	0.06	0.30	0.08	-0.05	0.08	0.12	0.86	-0.29	-0.12	-0.01	0.01	0.04	1.00	-0.07	-0.07	0.85	-0.40	-0.06	0.10	-0.01	-0.03	0.27	0.12	-0.03	-0.07	-0.03
A21	-0.17	-0.12	-0.06	-0.07	0.45	-0.15	-0.04	0.23	0.23	0.16	-0.02	-0.05	-0.09	-0.09	-0.07	0.51	-0.08	-0.09	0.13	-0.07	1.00	-0.04	-0.14	-0.12	-0.16	0.22	0.04	-0.06	-0.05	-0.04	0.09	0.56	0.41
A22	0.08	-0.11	-0.06	-0.06	-0.25	0.67	0.01	0.13	-0.07	0.08	-0.09	-0.03	-0.05	-0.08	-0.05	-0.17	0.23	-0.11	0.07	-0.07	-0.04	1.00	-0.13	-0.11	0.34	0.27	-0.17	0.16	0.04	0.05	0.13	-0.05	-0.03
A23	-0.10	0.95	-0.28	-0.29	-0.21	-0.11	0.16	-0.04	0.23	0.01	-0.01	0.04	0.10	0.81	-0.31	-0.15	-0.03	0.12	-0.07	0.85	-0.14	-0.13	1.00	-0.39	-0.02	-0.10	0.02	-0.04	0.11	0.36	-0.06	-0.12	-0.01
A24	-0.06	-0.42	0.61	-0.41	-0.24	-0.11	0.06	0.10	0.17	0.14	0.13	0.03	0.08	-0.37	0.65	-0.20	-0.03	0.08	-0.02	-0.40	-0.12	-0.11	-0.39	1.00	0.05	0.00	0.05	-0.07	-0.08	-0.14	-0.02	-0.11	-0.02
A25	-0.07	-0.04	-0.04	-0.05	-0.23	0.57	-0.03	-0.05	-0.13	0.02	0.10	0.02	0.06	-0.05	-0.05	-0.18	0.38	0.11	-0.14	-0.06	-0.16	0.34	-0.02	-0.05	1.00	-0.17	-0.08	0.11	-0.02	-0.01	-0.17	-0.14	-0.01
A26	0.22	-0.05	0.03	0.05	-0.01	0.05	0.16	0.31	0.01	0.26	-0.17	0.04	0.02	0.01	0.13	0.04	0.00	-0.31	0.33	0.10	0.22	0.27	-0.10	0.00	-0.17	1.00	-0.11	0.06	0.14	0.15	0.26	0.19	-0.11
A27	-0.13	0.01	0.05	0.00	0.09	-0.19	-0.07	-0.07	0.03	-0.06	0.22	0.02	0.02	0.00	0.01	0.05	-0.08	-0.18	-0.15	-0.01	0.04	-0.17	0.02	0.05	-0.08	-0.11	1.00	0.43	-0.18	-0.19	-0.05	-0.01	0.07
A28	0.16	-0.04	-0.02	-0.07	-0.08	0.21	-0.06	0.06	-0.09	0.06	-0.16	0.01	-0.02	-0.02	-0.04	-0.07	0.10	-0.14	-0.22	-0.03	-0.06	0.16	-0.04	-0.07	0.11	0.06	0.43	1.00	-0.09	-0.06	0.04	-0.01	-0.05
A29	0.16	0.19	-0.13	0.04	-0.13	0.00	0.18	0.15	0.07	0.16	-0.07	0.14	0.18	0.37	-0.02	-0.08	0.03	0.01	-0.01	0.27	-0.05	0.04	0.11	-0.08	-0.02	0.14	-0.18	-0.09	0.10	0.30	0.03	0.00	-0.03
A30	0.17	0.10	-0.12	0.06	-0.11	0.00	0.12	0.13	0.08	0.14	-0.12	0.13	0.16	0.10	0.24	-0.06	0.01	-0.01	-0.01	0.12	-0.04	0.05	0.09	-0.14	-0.01	0.15	-0.19	-0.09	0.30	1.00	0.03	0.00	-0.07
A31	0.11	-0.06	0.00	-0.01	0.09	0.01	0.05	0.22	0.09	0.17	-0.14	-0.04	-0.10	-0.02	0.02	0.06	-0.06	-0.02	0.01	-0.03	0.09	0.13	-0.06	-0.02	-0.17	0.26	-0.05	0.04	0.03	0.03	1.00	0.20	-0.11
A32	0.05	-0.11	-0.06	-0.07	0.38	-0.13	0.05	0.34	0.17	0.31	-0.07	0.02	0.03	-0.07	-0.05	0.43	-0.02	-0.03	0.04	0.07	0.59	-0.05	-0.12	-0.11	-0.14	0.19	-0.01	-0.01	0.00	0.00	0.20	1.00	-0.33
A33	-0.32	-0.01	-0.02	-0.02	0.17	-0.05	-0.16	-0.08	0.06	-0.14	0.06	-0.10	-0.14	-0.03	-0.04	0.11	-0.07	0.02	0.02	-0.03	0.41	-0.05	-0.01	-0.02	-0.01	-0.11	0.07	-0.05	-0.08	-0.07	-0.11	-0.35	1.00
A34	-0.03	-0.08	-0.06	-0.07	0.33	-0.10	-0.07	-0.04	0.05	0.01	0.01	-0.04	-0.07	-0.06	-0.08	0.29	-0.04	-0.01	0.01	-0.08	0.22	-0.08	-0.08	-0.10	-0.09	-0.04	0.03	-0.04	-0.03	0.00	0.26	0.07	0.00
A35	0.51	0.00	0.01	-0.03	-0.04	0.01	0.21	0.08	-0.02	0.15	-0.11	0.10	0.19	0.01	0.01	-0.01	0.06	-0.03	-0.01	0.00	-0.07	0.01	-0.02	-0.03	-0.08	0.00	-0.02	0.01	0.02	0.02	0.04	0.02	-0.16
A36	0.58	-0.08	-0.03	0.00	-0.05	0.02	0.20	0.08	-0.05	0.16	-0.24	0.15	0.20	-0.04	0.03	-0.02	0.04	-0.04	0.01	-0.02	-0.12	0.08	-0.08	-0.03	-0.02	0.24	-0.12	0.18	0.15	0.15	0.09	0.05	-0.23
A37	0.32	-0.06	0.02	0.03	-0.03	0.01	0.20	0.00	-0.03	0.06	0.08	0.13	0.16	-0.05	0.03	-0.02	0.03	-0.01	-0.02	-0.06	-0.09	0.00	-0.06	0.03	-0.01	0.00	0.02	0.00	-0.02	0.00	0.00	0.00	-0.15
A38	-0.31	-0.02	0.05	0.04	0.03	-0.01	-0.07	-0.07	0.01	-0.09	0.22	-0.01	-0.01	-0.03	0.03	0.00	-0.02	0.04	-0.02	-0.04	0.03	-0.05	0.00	0.08	0.03	-0.11	0.08	-0.07	-0.14	-0.12	-0.06	-0.03	0.13
A39	0.22	-0.03	0.01	0.01	-0.01	0.00	0.11	-0.01	-0.02	0.03	-0.03	0.03	0.08	-0.03	0.02	-0.01	0.02	-0.01															

	A34	A35	A36	A37	A38	A39	A40	A41	A42	A43	A44	A45	A46	A47	A48	A49	A50	A51	A52	A53	A54	A55	A56	A57	A58	A59	A60	A61	A62	A63	A64	A65	class
A01	-0.03	0.51	0.58	0.32	-0.31	0.22	0.22	0.30	0.06	0.24	0.72	0.30	0.37	0.30	0.22	0.23	0.10	0.13	0.25	0.29	0.19	0.18	0.26	0.37	0.20	0.14	0.65	0.06	-0.01	-0.06	-0.01	-0.10	0.34
A02	-0.08	0.00	-0.08	-0.06	-0.02	-0.03	-0.04	-0.07	0.00	-0.08	-0.04	-0.08	-0.01	-0.06	-0.04	-0.01	-0.01	-0.02	-0.06	-0.02	-0.05	-0.02	-0.01	0.04	0.03	0.02	-0.04	-0.09	-0.19	0.92	-0.41	0.03	-0.02
A03	-0.06	0.01	-0.03	0.02	0.05	0.01	0.01	0.03	-0.01	0.00	-0.01	0.12	0.15	0.12	0.09	0.12	0.09	0.07	0.04	0.07	0.04	0.04	0.07	0.10	0.09	0.08	-0.05	-0.04	-0.11	-0.27	0.57	0.02	0.07
A04	-0.07	-0.03	0.00	0.03	0.04	0.01	0.02	0.03	0.00	-0.01	-0.02	0.11	0.15	0.11	0.07	0.09	0.04	0.06	0.08	0.12	0.08	0.07	0.08	0.12	0.08	0.08	-0.05	-0.04	-0.11	-0.28	0.57	0.00	0.08
A05	0.33	-0.04	-0.05	-0.03	0.03	-0.01	-0.01	-0.05	-0.01	0.00	-0.07	0.12	0.13	0.12	0.04	0.06	0.09	0.06	-0.03	0.01	0.01	-0.01	-0.05	-0.06	-0.05	0.00	-0.05	-0.24	0.71	-0.20	-0.23	-0.03	0.10
A06	-0.10	0.01	0.02	0.01	-0.01	0.00	0.00	0.01	0.00	0.01	0.01	0.15	-0.01	0.15	0.10	0.02	0.08	0.10	0.10	-0.01	0.08	0.10	0.10	0.02	0.08	0.10	0.03	0.70	-0.24	-0.09	-0.08	0.02	0.14
A07	-0.07	0.21	0.20	0.20	-0.07	0.11	0.13	0.59	0.04	0.19	0.29	0.31	0.41	0.31	0.24	0.26	0.10	0.12	0.29	0.34	0.18	0.18	0.38	0.40	0.25	0.21	0.07	-0.01	0.03	0.23	0.11	0.10	0.34
A08	-0.04	0.08	0.08	0.00	-0.07	-0.01	0.02	0.08	0.01	0.00	0.10	0.85	0.73	0.85	0.55	0.50	0.43	0.62	0.51	0.43	0.34	0.56	0.53	0.42	0.41	0.84	0.25	0.46	0.51	0.11	0.32	0.20	0.53
A09	0.05	-0.02	-0.05	-0.03	0.01	-0.02	-0.01	-0.01	0.01	-0.05	-0.04	0.37	0.41	0.37	0.23	0.25	0.17	0.26	0.22	0.25	0.16	0.28	0.26	0.26	0.21	0.32	-0.02	-0.05	0.23	0.28	0.26	-0.04	0.34
A10	0.01	0.15	0.16	0.06	-0.09	0.03	0.09	0.17	0.03	0.08	0.19	0.87	0.81	0.87	0.53	0.52	0.43	0.55	0.47	0.46	0.36	0.50	0.57	0.54	0.49	0.81	0.18	0.42	0.46	0.13	0.31	0.19	0.56
A11	0.01	-0.11	-0.24	0.08	0.22	-0.03	0.04	0.58	0.04	0.11	-0.18	-0.06	-0.01	-0.06	-0.01	0.03	0.00	-0.16	-0.07	-0.03	-0.06	-0.20	0.09	0.05	0.02	-0.13	-0.51	-0.03	-0.01	-0.03	0.05	0.23	-0.09
A12	-0.04	0.10	0.15	0.13	-0.01	0.03	0.04	0.14	0.09	0.03	0.17	0.21	0.25	0.21	0.14	0.16	0.09	0.08	0.17	0.18	0.10	0.10	0.50	0.51	0.32	0.20	0.06	0.08	0.00	0.11	0.08	0.07	0.14
A13	-0.07	0.19	0.20	0.16	-0.01	0.08	0.13	0.22	0.08	0.05	0.26	0.33	0.38	0.33	0.22	0.26	0.18	0.15	0.23	0.26	0.17	0.16	0.54	0.56	0.46	0.28	0.08	0.10	-0.02	0.19	0.13	0.19	0.21
A14	-0.06	0.01	-0.04	-0.05	-0.03	-0.03	-0.02	-0.03	0.00	-0.07	-0.01	0.06	0.11	0.06	0.03	0.06	0.03	0.06	0.04	0.06	0.01	0.07	0.10	0.13	0.09	0.11	0.00	-0.06	-0.14	0.96	-0.35	0.06	0.07
A15	-0.05	0.01	0.03	0.03	0.03	0.02	0.04	0.07	0.00	-0.01	0.03	0.30	0.35	0.30	0.19	0.24	0.14	0.18	0.20	0.23	0.14	0.18	0.22	0.26	0.19	0.19	0.00	-0.04	-0.10	-0.29	0.81	0.06	0.18
A16	0.29	-0.01	-0.02	-0.02	0.00	-0.01	-0.01	-0.02	-0.01	0.01	-0.03	0.14	0.15	0.14	0.05	0.06	0.08	0.08	0.02	0.04	0.04	0.04	0.11	0.10	0.11	0.18	0.01	-0.16	0.58	-0.13	-0.17	-0.07	0.12
A17	-0.04	0.06	0.04	0.03	-0.02	0.02	0.02	0.06	0.01	0.00	0.07	0.29	0.15	0.29	0.18	0.13	0.15	0.18	0.16	0.08	0.13	0.15	0.43	0.30	0.42	0.40	0.05	0.62	-0.09	-0.01	-0.04	0.10	0.14
A18	-0.01	-0.03	-0.04	-0.01	0.04	-0.01	0.01	0.00	0.01	-0.05	-0.04	0.03	0.03	0.03	0.00	0.03	0.07	-0.01	-0.02	0.00	0.01	-0.05	0.34	0.07	0.15	-0.01	-0.11	-0.04	-0.04	0.06	0.01	0.50	-0.07
A19	0.01	-0.01	0.01	-0.02	-0.02	0.00	-0.02	-0.04	-0.03	0.01	0.00	-0.02	0.01	-0.02	0.00	-0.03	-0.05	0.02	0.01	0.02	0.00	0.04	-0.05	-0.07	-0.14	0.01	0.07	-0.01	0.04	-0.03	0.02	-0.49	0.08
A20	-0.06	0.00	-0.02	-0.06	-0.04	-0.03	-0.03	-0.05	0.00	-0.05	-0.02	0.01	0.06	0.01	-0.01	0.00	-0.04	0.02	0.06	0.08	0.01	0.09	0.05	0.08	0.03	0.09	0.02	-0.07	-0.14	0.90	-0.37	-0.04	0.09
A21	0.22	-0.07	-0.12	-0.09	0.03	-0.05	0.02	-0.10	-0.02	-0.06	-0.13	0.15	0.12	0.15	0.04	0.01	0.00	0.10	0.13	0.10	0.07	0.19	0.06	0.00	-0.01	0.18	-0.04	-0.08	0.55	-0.09	-0.07	-0.14	0.17
A22	-0.06	0.01	0.08	0.00	-0.05	0.00	0.00	0.02	-0.01	0.04	0.04	0.13	0.01	0.13	0.09	-0.05	-0.12	0.10	0.30	0.14	0.19	0.31	0.12	0.00	-0.02	0.16	0.13	0.47	-0.12	-0.08	-0.04	-0.11	0.23
A23	-0.08	-0.02	-0.08	-0.08	0.00	-0.03	0.04	-0.08	0.01	-0.08	-0.06	-0.11	-0.06	-0.11	-0.10	-0.07	-0.04	-0.08	-0.07	-0.03	-0.05	-0.04	-0.04	0.01	0.02	-0.02	-0.08	-0.10	-0.20	-0.04	-0.39	0.07	-0.06
A24	-0.10	-0.03	-0.03	0.03	0.08	0.01	0.02	0.04	0.00	-0.01	-0.04	0.12	0.17	0.12	0.07	0.11	0.07	0.06	0.05	0.12	0.09	0.04	0.08	0.14	0.13	0.06	-0.11	-0.07	-0.18	-0.39	0.78	0.03	0.07
A25	-0.09	-0.06	-0.02	-0.01	0.03	-0.02	0.01	0.02	0.01	-0.01	-0.05	-0.11	-0.12	-0.11	-0.28	-0.20	-0.14	-0.29	-0.03	-0.01	0.10	-0.05	-0.34	0.00	0.11	-0.09	-0.12	0.28	-0.21	-0.05	-0.08	0.14	-0.05
A26	-0.04	0.00	0.24	0.00	-0.11	0.00	0.05	0.10	0.00	0.12	0.12	0.35	0.35	0.35	0.17	0.06	-0.10	0.21	0.58	0.47	0.25	0.60	0.23	0.15	-0.02	0.32	0.28	0.13	0.23	0.01	0.15	-0.42	0.51
A27	0.03	-0.02	-0.12	0.02	0.08	0.00	0.00	0.06	0.01	-0.03	-0.07	0.08	-0.04	-0.08	-0.05	-0.01	-0.02	-0.07	-0.09	-0.05	-0.04	-0.13	-0.03	-0.01	-0.04	-0.09	-0.17	-0.15	0.05	0.00	0.02	-0.08	-0.13
A28	-0.04	0.01	0.16	0.00	-0.07	0.01	0.01	0.01	0.00	0.03	0.10	0.07	0.03	0.07	0.04	0.01	0.02	0.03	0.06	0.04	0.04	0.09	0.03	0.03	0.04	0.05	0.16	0.17	-0.04	-0.03	-0.05	0.05	0.09
A29	-0.03	0.02	0.15	-0.02	-0.14	-0.04	0.03	0.04	0.02	0.03	0.11	0.18	0.18	0.16	0.11	0.09	0.03	0.11	0.17	0.18	0.11	0.17	0.14	0.15	0.10	0.12	0.17	0.02	-0.05	-0.03	-0.04	0.01	0.18
A30	-0.03	0.02	0.15	0.00	-0.12	-0.02	0.03	0.02	0.01	0.03	0.12	0.14	0.17	0.14	0.08	0.07	0.00	0.09	0.18	0.18	0.10	0.19	0.12	0.13	0.07	0.11	0.19	0.00	-0.03	0.10	0.39	0.02	0.14
A31	0.00	0.04	0.09	0.00	-0.06	0.02	0.01	0.00	0.00	0.04	0.07	0.23	0.20	0.23	0.29	0.20	0.12	0.31	0.26	0.20	0.09	0.28	0.05	0.02	-0.04	0.13	0.16	0.05	0.21	-0.03	0.01	0.03	0.24
A32	0.26	0.02	0.05	0.00	-0.03	0.03	0.01	0.01	0.00	0.07	0.01	0.30	0.30	0.30	0.23	0.21	0.15	0.27	0.19	0.17	0.09	0.22	0.16	0.13	0.08	0.25	0.10	-0.04	0.57	-0.08	-0.06	-0.02	0.25
A33	0.07	-0.16	-0.20	-0.15	0.10	-0.12	0.02	-0.17	-0.02	-0.21	-0.21	-0.15	-0.19	-0.15	-0.15	-0.15	-0.08	-0.12	-0.14	-0.15	-0.07	-0.11	-0.14	-0.17	-0.11	-0.10	-0.20	-0.07	0.05	-0.01	-0.04	-0.01	-0.15
A34	1.00	-0.02	-0.02	-0.01	0.01	0.00	0.00	0.01	-0.04	0.00	-0.02	0.02	-0.02	-0.02	-0.01	0.01	-0.01	-0.06	-0.04	-0.05	-0.08	-0.04	-0.05	-0.06	-0.02	-0.01	-0.05	0.10	-0.06	-0.08	-0.01	0.03	0.03
A35	-0.02	1.00	-0.25	0.27	-0.18	0.22	0.04	0.17	0.05	0.13	0.39	0.17	0.20	0.17	0.29	0.37	0.27	0.17	-0.07	-0.10	-0.10	-0.04	0.13	0.19	0.13	0.07	0.33	0.04	-0.02	0.00	-0.02	0.01	0.13
A36	-0.02	-0.25	1.00	0.05	-0.22	0.04	0.18	0.15	0.02	0.13	0.30	0.18	0.22	0.18	-0.03	-0.08	-0.13	0.00	0.34	0.41	0.30	0.24	0.17	0.23	0.11	0.10	0.38	0.04	0.01	-0.06	0.01	-0.09	0.26
A37	-0.01	0.27	0.05	1.00	0.42	0.60	-0.01	0.21	0.07	0.19	0.53	0.04	0.10	0.04	0.08	0.11	0.08	0.02	0.02	0.04	0.01	-0.01	0.08	0.12	0.05	-0.01	0.12	0.01	-0.03	-0.06	0.01	0.00	0.06
A38	0.01	-0.18	-0.22	0.42	1.00	0.10	-0.02	-0.01	0.01	0.00	0.02	-0.12	-0.13	-0.12	-0.08	-0.07	-0.04	-0.08	-0.10	-0.10	-0.08	-0.09	-0.09	-0.11	-0.06	-0.08	-0.29	-0.03	-0.01	-0.03	0.03	0.06	-0.13
A39	0.00	0.22	0.04	0.60	0.10	1.00	-0.10	0.13	-0.02	0.14	0.23	0.02	0.05	0.02	0.06	0.07	0.04	0.03	-0.01	-0.01	-0.02	-0.01	0.01	0.02	0.01	0.00	0.09	0.00	-0.02	-0.03	0.00		