BayesChess: Programa de Ajedrez Adaptativo Basado en Redes Bayesianas*

Antonio Fernández Álvarez and Antonio Salmerón Cerdán

Depto. Estadística y Matemática Aplicada
Universidad de Almería
afa109@alboran.ual.es, Antonio.Salmeron@ual.es

Resumen En este trabajo presentamos un programa de ajedrez capaz de adaptar su estrategia al usuario al que se enfrenta y de refinar la función de evaluación que guía el proceso de búsqueda en base a su propia experiencia de juego. La capacidad adaptativa y de aprendizaje se ha implementado mediante redes bayesianas. Mostramos el proceso de aprendizaje del programa mediante una experimentación consistente en una serie de partidas que evidencian una mejora en los resultados después de la fase de aprendizaje.

1. Introducción

Las redes bayesianas se han mostrado en los últimos años como una herramienta adecuada para la modelización de situaciones en las que interviene un gran número de variables y existe incertidumbre asociada al valor de las mismas en un momento dado [5,7]. Uno de los problemas en los que está cobrando especial importancia el uso de redes bayesianas es en la *clasificación* o reconocimiento de patrones, que consiste en predecir la clase a la que pertenece un objeto dado que se conoce el valor de algunos atributos del mismo. El problema de la clasificación está presente en la construcción de sistemas que se adapten al usuario, desde el momento en que el sistema tiene que determinar de qué tipo de usuario se trata y actuar en consecuencia.

En este trabajo describimos un programa de ajedrez capaz de adaptarse al usuario y ajustar su estrategia de juego según el estilo del usuario, siendo además capaz de aprender de su propia experiencia en el sentido de refinar la función de evaluación o heurística de búsqueda que se emplea para explorar el árbol de jugadas. Esta funcionalidad adaptativa y de aprendizaje automático se ha implementado usando redes bayesianas. Más concretamente, hemos empleado redes bayesianas orientadas a la clasificación, basándonos en la estructura *Naive Bayes*, especialmente adecuada en problemas en los que interviene un gran número de variables y la base de datos de aprendizaje es de tamaño limitado [6]. Hemos adoptado el nombre BayesChess por ser su particularidad el empleo de redes bayesianas.

* Trabajo subvencionado por el Ministerio de Educación y Ciencia, proyecto TIN2004-06204-C03-01 y por fondos FEDER

El objetivo no es conseguir un programa de ajedrez realmente competitivo con programas de la talla de Deep Blue [9] o Fritz [8], sino comprobar la viabilidad de construir sistemas adaptativos empleando redes bayesianas. Hemos elegido el ajedrez por considerar que éste presenta una serie de características que hacen apropiado el empleo de sistemas adaptativos:

- El programa interactúa constantemente con el usuario y ha de responder a las acciones del mismo.
- La imposibilidad de calcular todas las jugadas posibles hacen necesario el uso de una heurística.
- La validez de la heurística empleada puede contrastarse en base a los resultados obtenidos.
- Existen diferentes estilos de juego o estrategias que tanto el usuario como el programa pueden adoptar.

Por tanto, el objetivo de este trabajo es el uso de redes bayesianas para dotar de adaptatividad a un programa de ajedrez. En ese sentido, nos hemos concentrado en lo siguiente:

1. Refinamiento de la heurística de búsqueda, en base a la experiencia de juego del programa, mediante una red bayesiana.
2. Uso de una red bayesiana para clasificar el comportamiento del usuario y adoptar una estrategia en función del mismo.

El resto del trabajo se organiza como sigue: en la sección 2 revisamos algunos conceptos de redes bayesianas y clasificación. Continuamos exponiendo el diseño del motor de juego y la heurística de búsqueda en la sección 3. La actualización automática de dicha heurística se explica en la sección 4, mientras que la adaptación al comportamiento del usuario es el objeto de la sección 5. La experimentación llevada a cabo para evaluar el proceso de aprendizaje se describe en la sección 6, y el trabajo finaliza con las conclusiones en la sección 7.

2. Redes bayesianas y clasificación

Consideremos un problema caracterizado por un conjunto de variables $\mathbf{X} = \{X_1, \ldots, X_n\}$. Una red bayesiana [7,12] es un grafo dirigido acíclico donde cada nodo representa una variable del problema, y tiene asignada una distribución de probabilidad condicionada a sus padres. La presencia de un arco entre dos variables expresa la existencia de dependencia entre ambas, cuantificada por la distribución de probabilidad asignada a los nodos. En términos computacionales, una importante propiedad de las redes bayesianas es que la distribución conjunta sobre todas las variables de la red se factoriza de acuerdo con el concepto de *d*-separación [12] como sigue:

$$p(x_1, \ldots, x_n) = \prod_{i=1}^{n} p(x_i | pa(x_i)) \ , \tag{1}$$

donde $Pa(X_i)$ denota el conjunto de padres de la variable X_i y $pa(x_i)$ una configuración concreta de valores de los mismos. Esta factorización implica que se puede especificar la distribución conjunta sobre todas las variables de la red con un importante ahorro de espacio. Por ejemplo, la distribución conjunta sobre las variables de la red de la figura 1, suponiendo que todas las variables son binarias, requeriría almacenar $2^5 - 1 = 31$ valores, mientras que haciendo uso de la factorización se puede representar la misma información usando sólo 11 valores (ver cuadro 1).

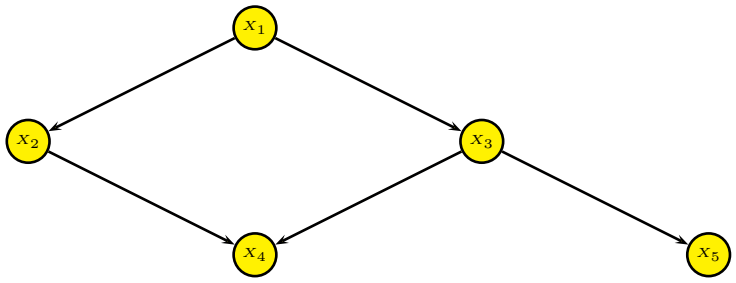

Figura 1. Ejemplo de red Bayesiana

Cuadro 1. Ejemplo de distribución factorizada para la red de la figura 1

$p(X_1 = 0) = 0,20$	$p(X_2 = 0\|X_1 = 0) = 0,80$
$p(X_2 = 0\|X_1 = 1) = 0,80$	$p(X_3 = 0\|X_1 = 0) = 0,20$
$p(X_3 = 0\|X_1 = 1) = 0,05$	$p(X_4 = 0\|X_2 = 0, X_3 = 0) = 0,80$
$p(X_4 = 0\|X_2 = 1, X_3 = 0) = 0,80$	$p(X_4 = 0\|X_2 = 0, X_3 = 1) = 0,80$
$p(X_4 = 0\|X_2 = 1, X_3 = 1) = 0,05$	$p(X_5 = 0\|X_3 = 0) = 0,80$
$p(X_5 = 0\|X_3 = 1) = 0,60$	

Una red bayesiana puede usarse como clasificador si está compuesta por una variable clase, C, y un conjunto de variables predictoras $X_1, \ldots, X_n$, de forma que un individuo con características observadas $x_1, \ldots, x_n$ será clasificado como perteneciente a la clase c^* obtenida como

$$c^* = \arg \max_{c \in \Omega_C} p(c|x_1, \ldots, x_n) \ , \tag{2}$$

donde Ω_C denota el conjunto de posibles valores de la variable clase C.

Obsérvese que $p(c|x_1, \ldots, x_n)$ es proporcional a $p(c) \times p(x_1, \ldots, x_n|c)$, y por tanto, resolver el problema de clasificación requeriría especificar una distribución sobre las n variables predictoras para cada valor de la clase. El coste asociado

puede ser muy elevado. Sin embargo, usando la factorización determinada por la red este coste se reduce. El caso extremo es el clasificador conocido como *Naive Bayes* (ver, por ejemplo [3]), que supone que todas las variables predictoras son independientes entre sí dada la clase. Esto se traduce en una estructura como la representada en la figura 2.

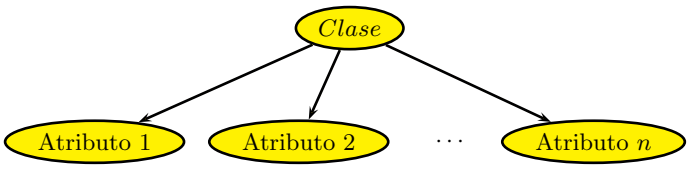

Figura 2. Topología de un clasificador Naïve Bayes

La fuerte suposición de independencia que hace este modelo se compensa con el reducido número de parámetros que hay que estimar, dado que en este caso se verifica que

$$p(c|x_1, \ldots, x_n) = p(c) \prod_{i=1}^{n} p(x_i|c) \ . \tag{3}$$

3. Diseño del motor de juego de ajedrez

El juego del ajedrez ha sido estudiado en profundidad por la Inteligencia Artificial, dentro de los llamados *juegos de información completa* (ver, por ejemplo, [11]). En este trabajo hemos considerado un motor de juego basado en el algoritmo mini-max con poda alfa-beta. Existen refinamientos en los algoritmos de búsqueda orientados a ajedrez, pero quedan fuera del ámbito de este trabajo. Sí es relevante la heurística utilizada para guiar el proceso de búsqueda pues, como describiremos más adelante, será actualizada mediante una red bayesiana aprendida en base a la experiencia de juego del programa.

La heurística que hemos considerado se basa en dos aspectos: material (piezas de cada jugador en el tablero) y situación de cada pieza (dependiendo de la casilla que ocupen en un momento dado, las piezas pueden ser más o menos valiosas). De forma adicional, hemos dado importancia también al hecho de dar jaque al rey adversario.

La evaluación del material se realiza asignando una puntuación a cada pieza. Hemos elegido la puntuación habitual en programas de ajedrez, mostrada en el cuadro 2. El rey no tiene puntuación, pues no es necesario al estar prohibida la captura del mismo.

En cuanto a la valoración de la posición de cada pieza, hemos empleado una matriz de 8×8 para cada ficha, de forma que cada celda contiene la puntuación

Cuadro 2. Puntuación de las piezas en la heurística empleada

Pieza	Peón	Alfil	Caballo	Torre	Dama
Puntuación	100	300	300	500	900

añadida al valor de la heurística en caso de que la pieza esté situada en ella. En el cuadro 3 puede verse un ejemplo de estas matrices, para el caso del caballo blanco. Nótese como se favorece la colocación del caballo en casillas centrales, donde tiene mayor capacidad de acción.

Cuadro 3. Pesos asociados a la posición del caballo blanco

-10	-10	-10	-10	-10	-10	-10	-10
-10	0	0	0	0	0	0	-10
-10	0	5	5	5	5	0	-10
-10	0	5	10	10	5	0	-10
-10	0	5	10	10	5	0	-10
-10	0	5	5	5	5	0	-10
-10	0	0	0	0	0	0	-10
-10	-30	-10	-10	-10	-10	-30	-10

En total, la función heurística está definida por 838 parámetros ajustables, que son el valor de cada pieza, el valor de dar jaque y el número almacenado en cada celda de cada una de las matrices 8×8 definidas anteriormente.

4. Actualización automática de la heurística

En esta sección describimos el proceso de aprendizaje, o más precisamente de refinamiento de los parámetros de la heurística descrita en la sección 3. Con el auge de las técnicas de aprendizaje automático en los años 80, se consideró la aplicación de las mismas al ajedrez por computador, concluyéndose que sólo serían aplicables de forma marginal, como vías de extracción de patrones de libros de aperturas [13]. Sin embargo, más adelante surgieron aplicaciones de técnicas de clasificación, principalmente para la evaluación de posiciones de la fase final del juego [4].

Para el ajuste de los parámetros de la heurística, hemos considerado una red bayesiana con estructura tipo Naive Bayes con la salvedad de que en lugar de una variable clase hay dos: la *fase actual de la partida* (apertura, medio juego o final) y el *resultado de la partida* (ganar, perder, tablas). Como variables predictoras, se han empleado todos los parametros ajustables de la heurística descrita en la sección 3, lo que significa que la red cuenta con un total de 776 variables con la estructura mostrada en la figura 3. El elevado número de variables viene dado principalmente porque hay una variable por cada una de las 64 posibles ubicaciones de cada una de las piezas en el tablero. La razón por la que se ha usado una estructura de red tipo Naive Bayes es precisamente el alto número

de variables, ya que el uso de una estructura más compleja aumentaría drásticamente el tiempo necesario para evaluarla, lo que ralentizaría la evaluación de las posiciones durante la exploración del árbol de búsqueda.

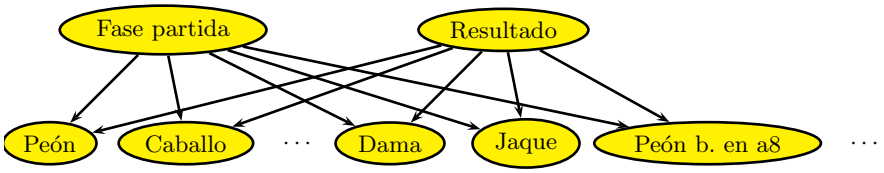

Figura 3. Estructura de la red bayesiana para el aprendizaje automático de la heurística

Los parámetros de la red bayesiana se estiman inicialmente a partir de una base de datos generada enfrentando a BayesChess contra él mismo, usando uno de los dos bandos la heurística tal y como se ha descrito anteriormente, y el otro una versión perturbada aleatoriamente, donde el valor de cada variable se incrementaba o decrementaba en un 20 %, 40 % o se mantenía a su valor inicial de forma aleatoria. En el cuadro 4 se puede ver el formato de dicha base de datos con unos casos de ejemplo. Vemos cómo para cada etapa de cada partida se ha generado una configuración de parámetros aleatorios que serán los que utilizará la heurística en la etapa concreta de la partida en juego. Al final de cada caso aparece el resultado de la misma. Evidentemente el resultado en cada *caso* de la base de datos perteneciente a una misma partida es el mismo, de ahí que aparezca repetido de forma consecutiva.

Cuadro 4. Ejemplo de casos de la base de datos de partidas

Fase partida	Peon	Caballo	Alfil	Torre	Dama	Jaque	Peón a8	Peon b8	...	Resultado
APERTURA	120	180	240	700	540	42	-6	0	...	PIERDEN
MEDIO	120	360	360	600	1260	36	3	0	...	PIERDEN
FINAL	120	420	180	400	720	42	-3	3	...	PIERDEN
APERTURA	140	360	420	300	1080	18	0	6	...	GANAN
MEDIO	100	300	360	500	1260	42	3	0	...	GANAN
FINAL	80	180	240	400	900	42	6	6	...	GANAN
APERTURA	120	420	420	400	720	18	-6	3	...	TABLAS
MEDIO	140	300	360	500	1260	42	3	0	...	TABLAS
FINAL	120	420	180	600	900	30	0	6	...	TABLAS

Cada una de las tablas de probabilidad en esta red bayesiana requiere la estimación de 45 valores, ya que para cada uno de los 5 posibles valores de una variable habrá que diferenciar la fase de la partida y el resultado conseguido. En el cuadro 5 podemos observar un ejemplo de la tabla de probabilidad condicio-

nada de la variable *Peón*. Se han usado las abreviaturas A, M y F para las fases de partida y G, P y T para el resultado.

Cuadro 5. Ejemplo de tabla de probabilidad de la variable *Peón*

	FASE PARTIDA	A	A	A	M	M	M	F	F	F
PEÓN	RESULTADO	G	P	T	G	P	T	G	P	T
	60	0,2	0,1	0,3	0,3	0,2	0,3	0,2	0,3	0,2
	80	0,3	0,1	0,1	0,1	0,2	0,1	0,2	0,1	0,2
	100	0,1	0,1	0,2	0,4	0,2	0,1	0,1	0,1	0,2
	120	0,1	0,2	0,4	0,1	0,3	0,1	0,3	0,2	0,3
	140	0,3	0,5	0,1	0,2	0,1	0,4	0,2	0,3	0,1

El proceso de aprendizaje del juego no tiene límite, ya que dependerá del número de partidas existente en la base de datos. Por tanto, cuantas más partidas se hayan jugado más se refinarán los parámetros de la heurística, y por tanto mejor será el juego de la máquina, como veremos en la sección 6. Una vez concluido el entrenamiento inicial, BayesChess puede adoptar la heurística aprendida y a partir de ahí refinarla con nuevas partidas, ya contra oponentes humanos.

Una vez construida la red bayesiana, BayesChess la utiliza para elegir los parámetros de la heurística. El proceso de selección es consiste en instanciar las dos variables clase (fase de partida y resultado) y a partir de ahí se obtiene la configuración de parámetros que maximiza la probabilidad de los valores instanciados de las variables *Fase de partida* y *Resultado*. Para poder conocer siempre en qué fase de partida se encuentra el juego, hemos considerado que la apertura la forman las 10 primeras jugadas y el final de juego es cuando no hay damas o el número de piezas es inferior a 10. En otro caso, entendemos que la partida está en fase de medio juego. En cuanto a la otra variable a instanciar (resultado) obviamente no se conoce en el momento del juego, pero se puede usar para determinar el estilo de juego de BayesChess. Por ejemplo, si instanciamos la variable resultado a *ganar*, elegirá la configuración de parámetros que maximizan la probabilidad de ganar, aunque ésta sea menor que la suma de las probabilidades de perder y hacer tablas. Esto puede ser equivalente a considerar que BayesChess adopta una estrategia agresiva. Por contra, puede optarse por minimizar la probabilidad de perder, o lo que es lo mismo, maximizar la de ganar o hacer tablas. Esto puede derivar en una estrategia de juego más conservadora. La forma de elegir estas configuraciones es mediante inferencia abductiva [2,10]. En el caso concreto de la red bayesiana empleada por BayesChess, el cálculo es sencillo, pues la configuración que maximiza la probabilidad de una instanciación dada, es la que se obtiene de maximizar cada tabla de probabilidad por separado para dicha instanciación, debido a la factorización expresada en la ecuación (3).

5. Adaptación a la situación del oponente

En esta sección se describe cómo hacer que la heurística aprendida se adapte, en cada momento, a la estrategia de juego del usuario al que se enfrenta. Para

ello hemos considerado tres posibles estrategias que puede adoptar el usuario: atacante, posicional y mixta.

Hemos implementado un clasificador Naïve Bayes para determinar la estrategia de juego de un usuario tomando como referencia una serie de características que serán las variables de dicho clasificador. Dichas características son: el primer movimiento, la situación de los enroques (opuestos o no), número de piezas más allá de la tercera fila (excepto peones) y número de peones avanzados sobre la posición del rey. La estructura del clasificador puede verse en la figura 4.

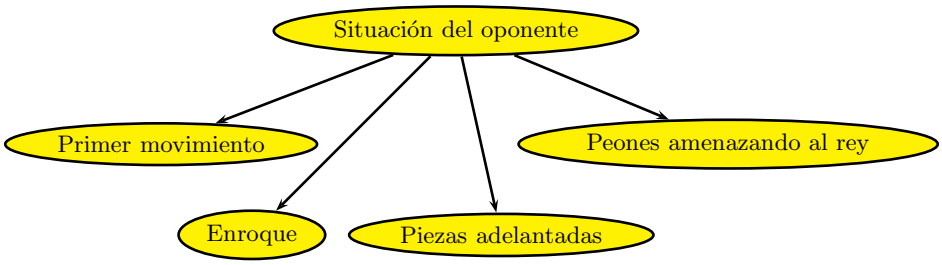

Figura 4. Estructura del clasificador de la estrategia del oponente

El entrenamiento del clasificador se ha realizado a partir de una base de datos de partidas de cuatro destacados jugadores profesionales conocidos por corresponderse con los tres estilos citados. En esas partidas, se han medido las variables anteriores y los valores obtenidos se han incluido en la base de datos de entrenamiento. En concreto, hemos seleccionado 2708 partidas de Robert Fischer y Gary Kasparov como modelo de juego atacante, 3078 de Anatoli Karpov como modelo de juego posicional o defensivo y, por último, 649 de Miguel Illescas como modelo de juego mixto. En el cuadro 6 se puede ver el formato de dicha base de datos con unos casos de ejemplo.

Cuadro 6. Ejemplo de casos de la base de datos de entrenamiento del clasificador de estrategias

1^{er} movimiento	Enroque	Piezas adelantadas	Peones amenazando rey	Estrategia
e4	iguales	2	1	Atacante
Cf6	opuestos	0	2	Atacante
Cf3	iguales	0	1	Mixta
d4	iguales	1	0	Defensiva
c5	iguales	0	0	Atacante
c4	opuestos	1	2	Defensiva
otro_n	iguales	1	1	Mixta

Usando este clasificador, BayesChess determina la estrategia del oponente instanciando las cuatro variables predictoras y calculando, para los valores

instanciados, cuál es el valor de la variable *Situación del oponente* con mayor probabilidad.

5.1. Proceso de adaptación al oponente

Una vez que determinado el tipo de juego que está desarrollando el oponente, BayesChess decide su propia estrategia usando la red bayesiana de ajuste de los parámetros de la heurística como sigue:

- Cuando la estrategia del oponente es **atacante**, elige aquellos parámetros que **minimizan la probabilidad de perder**.
- Cuando la estrategia del oponente es **defensiva**, elige aquellos parámetros que **maximizan la probabilidad de ganar**.
- Cuando la estrategia del oponente se clasifique como mixta, elige **aleatoriamente** una de las dos opciones anteriores.

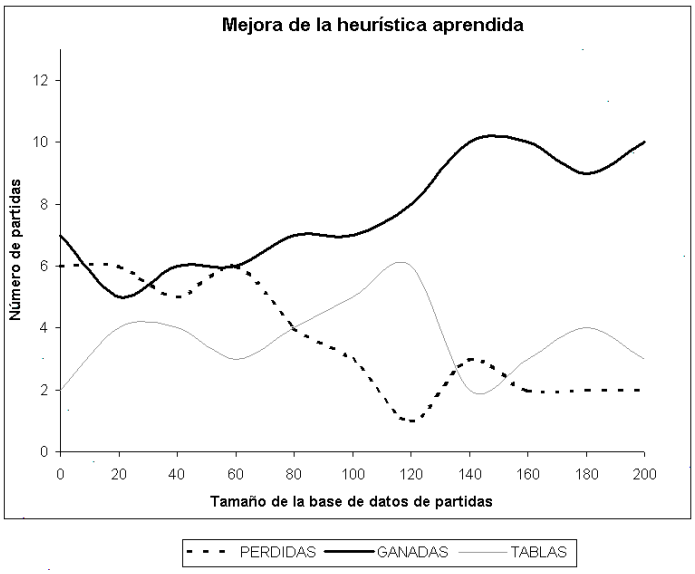

Figura 5. Evolución de los resultados entre la heurística aprendida y la fija

6. Experimentación

Hemos realizado dos experimentos para evaluar el proceso de aprendizaje de la heurística, en ambos casos usando una base de datos con 200 partidas jugadas entre la heurística inicial y una aleatoria.

El primer experimento consistió en realizar 7 torneos de 15 partidas entre BayesChess con la heurística fija y él mismo con la heurística aprendida con subconjuntos de más o menos partidas de la base de datos. En la figura 5 se observa cómo la heurística aprendida mejora sus resultados conforme crece el número de partidas jugadas.

El segundo experimento consistió en evaluar la puntuación asignada por la heurística a una posición determinada, concretamente a la mostrada en la figura 6, con distinto número de partidas en la base de datos. Se observa que en la posición de la figura 6, las blancas cuentan con un caballo y dos peones de ventaja, con lo que la valoración debe estar alrededor de -500 puntos (medidos desde el punto de vista de las negras). En la figura 7 puede observarse cómo efectivamente la heurística se acerca a dicho valor conforme crece el tamaño de la base de datos.

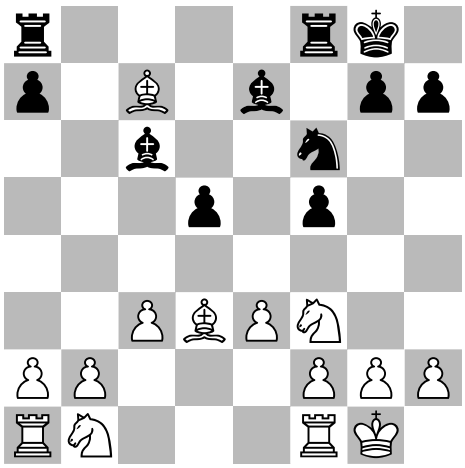

Figura 6. Tablero ejemplo para el segundo experimento

7. Conclusiones

En este hemos presentado BayesChess, un programa de ajedrez que adapta su comportamiento al enemigo al que se enfrenta y a su propia experiencia de juego. Los resultados de la experimentación indican que el aprendizaje conlleva una mejora práctica de los resultados y que la heurística ajusta sus parámetros hacia valores más precisos.

Pensamos que el uso de redes bayesianas aporta un valor añadido en la construcción de sistemas adaptables al usuario. En casos como BayesChess, donde el número de variables a tener en cuenta es muy alto, permiten hacer inferencias de forma eficiente utilizando topologías de red restringidas como el Naive Bayes.

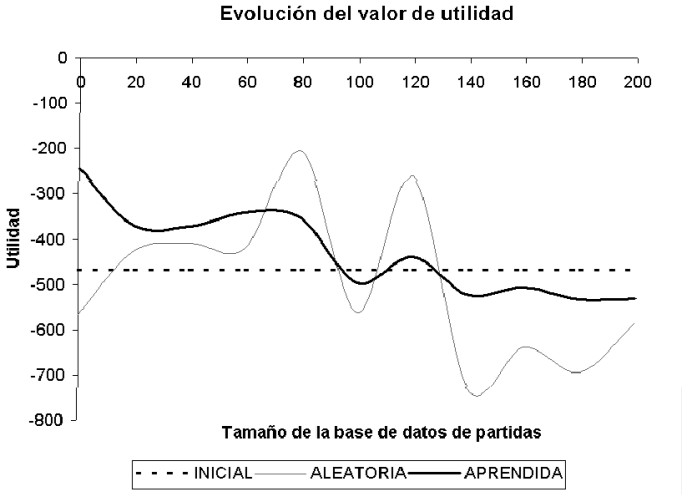

Figura 7. Evolución del valor de utilidad conforme aumenta el tamaño de la base de datos

No solamente el ajedrez, sino otros juegos de ordenador que requieran toma de decisiones por parte de la máquina, pueden beneficiarse del uso de redes bayesianas en la misma forma en que se propone en este trabajo. Un ejemplo inmediato son las damas, pero hay que tener en cuenta que en este caso la complejidad del juego es muy inferior, y por tanto la heurística, al menos a priori, no habrá de tener en cuenta tantas variables.

En un futuro esperamos mejorar el comportamiento puramente ajedrecístico de BayesChess refinando la implementación del algoritmo mini-max e introduciendo un nuevo clasificador que se emplee en los finales de juego. En este sentido, existen bases de datos con posiciones típicas de finales de torres y peones clasificadas como ganadoras, perdedoras o de tablas, que pueden ser utilizadas para entrenar el citado clasificador [1].

Referencias

1. C.L. Blake and C.J. Merz. UCI repository of machine learning databases. www.ics.uci.edu/∼mlearn/MLRepository.html, 1998. University of California, Irvine, Dept. of Information and Computer Sciences.
2. L.M. de Campos, J.A. Gámez, and S. Moral. Partial abductive inference in Bayesian networks by using probability trees. In *Proceedings of the 5th International Conference on Enterprise Information Systems (ICEIS'03)*, pages 83–91, Angers, 2003.
3. R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern classification.* Wiley Interscience, 2001.

4. J. Fürnkranz. Machine learning in computer chess: The next generation. *ICCA Journal*, 19(3):147–161, 1996.
5. J.A. Gámez, S. Moral, and A. Salmerón. *Advances in Bayesian networks*. Springer, Berlin, Germany, 2004.
6. J. Hernández, M.J. Ramírez, and C. Ferri. *Introducción a la minería de datos*. Pearson, 2004.
7. Finn V. Jensen. *Bayesian networks and decision graphs*. Springer, 2001.
8. K. Muller. The clash of the titans: Kramnik - FRITZ Bahrain. *IGCA Journal*, 25:233–238, 2002.
9. M. Newborn. *Kasparov vs. Deep Blue: Computer chess comes of age*. Springer-Verlag, 1997.
10. D. Nilsson. An efficient algorithm for finding the M most probable configurations in Bayesian networks. *Statistics and Computing*, 9:159–173, 1998.
11. N. Nilsson. *Inteligencia artificial: una nueva síntesis*. McGraw-Hill, 2004.
12. J. Pearl. *Probabilistic reasoning in intelligent systems*. Morgan-Kaufmann (San Mateo), 1988.
13. S.S. Skiena. An overview of machine learning in computer chess. *ICCA Journal*, 9(1):20–28, 1986.