CGO y COBLI: Clasificadores Oblicuos basados en Algoritmos Genéticos

José L. Álvarez Macías	José Riquelme Santos	Jacinto Mata Vázquez	José J. Mora Pérez
Dpto. I.E.S.I.A.	Dpto Leng. y Sist. Inf.	Dpto. I.E.S.I.A.	Dpto. I.E.S.I.A.
Universidad de Huelva	Universidad de Sevilla	Universidad de Huelva	Universidad de Huelva
alvarez@uhu.es	riquelme@lsi.us.es	mata@uhu.es	albeniz@uhu.es

RESUMEN:

En este artículo presentamos dos sistemas clasificadores basados en reglas mediante aprendizaje supervisado. CGO es un clasificador basado en un Algoritmo Genético (AG) que utiliza el C4.5 para determinar la bondad de su individuos y mostrar los resultados de la clasificación obtenida. Los individuos en CGO se obtienen como combinaciones lineales de los atributos originales. COBLI es un AG cuyos individuos representan una división en regiones del espacio a clasificar. La función de adecuación minimizará el error cometido en la clasificación, primando individuos que agrupan el mayor número de elementos bien clasificados y con menor número de regiones.

Palabras claves: Aprendizaje Automático, Algoritmos Genéticos, Árboles de Decisión Oblicuos.

ABSTRACT:

In this paper we present two classifier systems based on rules through supervised learning. CGO is a classifier based on a Genetic Algorithm (GA) that uses the C4.5 to determine the fitness of the individuals and to show the results of the obtained classification. The individuals in CGO are obtained as linear combinations from the original attributes. COBLI is a GA whose individuals represent a division in regions of the space to classify. The fitness function will minimize the assignment error in the classification, outweighing individual that group the greater number of well classified elements and with smaller number of regions.

Keyworks: Machine Learning, Genetic Algorithms, Obliques Decision Trees.

1.- Introducción

En la actualidad disponer de información digitalizada es fácil y barata de almacenar. El problema aparece cuando se desea gestionar. Estos datos en bruto rara vez aportan beneficios directamente. Su verdadero valor radica en la posibilidad de extraer información útil. En general, la capacidad del humano para analizar y comprender grandes cantidades de información decrece a medida que aumenta su capacidad para almacenarlos. Esto está produciendo un auge de técnicas de aprendizaje automático capaces de obtener un conjunto mínimo de reglas, inteligibles por el hombre, a partir de grandes volúmenes de datos; o sea, sistemas capaces de transformar información cuantitativa en información cualitativa.

Como punto de partida, este tipo de técnicas requiere información de elementos estructurada en atributos y la clasificación de éstos; y como resultado generará un conjunto de reglas para clasificar a todos los elementos. Este tipo de aprendizaje se conoce como aprendizaje supervisado.

De las diferentes técnicas de clasificación utilizadas, los sistemas basados en reglas, como los árboles de decisión, obtienen resultados muy prometedores. En ellos, cada regla representa una decisión en la clasificación. Cada decisión corresponde con un hiperplano del espacio a clasificar que lo divide en dos regiones. Los clasificadores se dividen en oblicuos y no oblicuos o paralelos. Para los clasificadores paralelos, cuando los valores de los atributos son numéricos, las decisiones son condiciones de la forma $Xi < C$; donde se compara un simple atributo Xi con una constante C. En los clasificadores oblicuos las decisiones están compuestas por condiciones sobre una combinación de varios atributos, de la forma $\Sigma AiXi<C$.

Entre las principales herramientas de clasificación están el C4.5 [1] y OC1 [2]. El C4.5 es un clasificador paralelo basado en el criterio de ganancia de información. El OC1 es un clasificador oblicuo basado en "deterministic hill climbing" y aleatoriedad. El C4.5, como clasificador paralelo, presentan soluciones aceptables, pero la necesidad de dividir el espacio a clasificar con hiperpanos ortogonales le puede llevar a generar un gran número de reglas. El OC1 soluciona en parte este problema al permitir hiperplanos oblicuos. El problema en el OC1 está en la necesidad de buscar el primer hiperplano, que al utilizar la estrategia "divide y vencerás", condiciona el resto de la clasificación. Por este motivo, el OC1 no se preocupa de encontrar la clasificación con menor número de rectas.

Para solucionar este problema, en este artículo se presentan dos trabajos sobre clasificadores basados en algoritmos genéticos. Por un lado, CGO es un clasificador oblicuo cuyo algoritmo principal es un AG que utiliza para la función de bondad la proporcionada por C4.5, una vez preparado éste con un individuo del AG; y COBLI es un clasificador oblicuo cuyo algoritmo principal es un AG donde la función de bondad minimizará errores e hiperplanos y maximizará la agrupación de puntos dentro de las regiones.

No todo son ventajas en los sistemas presentados, y es obvio pensar que presentan todas las desventajas propias de los AG's. Entre ellas, hay que destacar dos fundamentalmente: el elevado tiempo de computación y la aleatoriedad de las soluciones. Este último efecto provoca que la salida para una misma clasificación pueda ser diferente cada vez.

2. CGO

CGO es un clasificador oblicuo basado en AG y C4.5, mediante un método similar al usado en [3]. El algoritmo principal es un AG que necesita del C4.5 para obtener la función de bondad de cada individuo. De forma general, el sistema irá generando individuos a partir de los atributos iniciales y combinaciones de éstos y obtendrá los resultados de la clasificación a partir del C4.5. La mejor clasificación obtenida durante el paso de las generaciones será presentada como clasificación final de los elementos del conjunto.

2.1. Estructura de individuos en CGO

A la hora de aplicar el AG una decisión de vital importancia para obtener resultados interesantes es la definición de la estructura interna de los individuos en el algoritmo. En CGO un individuo representará combinaciones lineales de los atributos originales. La figura 1 muestra una tupla con la estructura de un individuo.

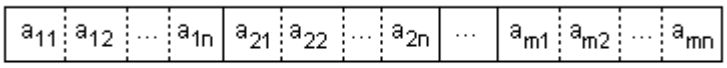

Fig.1. Representación interna de individuos CGO

Donde, para un caso de p atributos, a_{ij} son los coeficientes de las combinaciones de los atributos, n = p-1 y m es el número de combinaciones para ese individuo. El número de combinaciones de cada individuo es aleatorio y estará comprendido entre 0 (solo atributos originales) y el número máximo de combinaciones permitidas.

Para aclarar la estructura, un ejemplo con 3 atributos <X, Y, Z> donde se haya generado un individuo con dos combinaciones de atributos quedaría como muestra la figura 2. Donde la combinación de atributos que representa será: $X+a_{ij}Y+a_{ij+1}Z$. En adelante, por comodidad, se representará cada combinación con una letra mayúscula y su primer subíndice. Así, la figura 1 se representará según aparece en la figura 3.

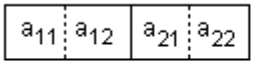

Fig.2. Ejemplo individuo CGO

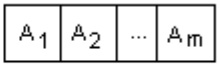

Fig.3. Representación abreviada de individuos

2.2. Algoritmo CGO.

El algoritmo principal de CGO es un AG. El resultado del algoritmo será el mejor individuo generado en el proceso de evolución. La salida producida será el conjunto de reglas generadas por el C4.5 para ese individuo y los valores de cada coeficiente. El criterio de terminación es establecido por un número máximo de generaciones.

La población inicial se genera mediante la inclusión de un primer individuo sin ninguna combinación de atributos (los atributos originales) y con la generación aleatoria tanto del número de combinaciones para cada individuo, entre 0 y el máximo de combinaciones, como de los coeficientes de las combinaciones, entre un máximo y un mínimo para el valor de los coeficientes, se genera el resto hasta completar el número de individuos establecido previamente. Los parámetros se especifican en el apartado 4.

El proceso de evolución se basa en un AG elitista: "el mejor individuo de cada población pasa a la siguiente". Un conjunto de nuevos individuos "sobrevivirán" para la siguiente generación, seleccionados aleatoriamente según su adaptación (función de bondad). El resto de individuos es generado mediante cruce de individuos existente.

El operador de cruce seleccionará dos individuos atendiendo a su adaptación, y producirá dos nuevos individuos que contendrán combinaciones de atributos de ambos individuos originales. Para ello, seleccionará en cada individuo padre un punto de corte e intercambiará los "genes" según la figura 4.

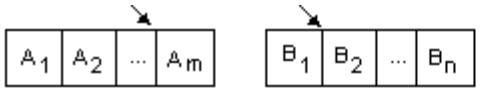

Fig.4.a) Individuos a cruzar por ↘

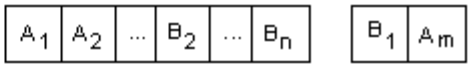

Fig 4.b) Individuos resultantes del cruce

Finalmente, los individuos pueden verse afectados por mutaciones atendiendo a una probabilidad. Las mutaciones van a consistir en el aumento o disminución, en una cantidad proporcional, de los coeficientes.

2.3. La función de bondad para CGO

La bondad o adaptación de un individuo se obtiene con la ayuda del C4.5. Así, para obtener la bondad de un individuo se prepara el fichero de entrenamiento con los atributos iniciales y la combinación de éstos indicada en el individuo y se ejecuta el C4.5 obteniéndose los datos bien clasificados, los errores y el número de reglas en la clasificación. Este valor es alterado con el número de combinaciones de atributos quedando la bondad de un individuo según (1). Con el fin de adecuar la bondad de cada individuo en forma de porcentaje, la ecuación (1) se divide por el total de la suma de bondades.

$$\text{bondad} = \text{aciertos} - \text{errores} - \text{nreglas} * 0{,}1 - \text{ncomb} * 0{,}01 \qquad (1)$$

3. COBLI

El algoritmo principal de COBLI es un AG cuyos individuos representan la división en regiones del espacio a clasificar. De forma general, los individuos están compuestos por un vector de coeficientes que representan los hiperplanos que delimitarán las regiones de la clasificación. Para determinar la bondad de cada clasificación el sistema analizará cada región determinando la clase mayoritaria de ésta, y así conocer el error generado. La función de bondad también primará clasificaciones que agrupen elementos del mismo tipo y con menor número de regiones.

La mejor clasificación obtenida determinará la clase de cada región. De esta forma, cada elemento queda clasificado según la región en la que se encuentra. La salida del algoritmo será un conjunto de reglas que clasifica por regiones los elementos.

3.1. Estructura de individuos en COBLI

La estructura de individuos en COBLI consiste en un vector de puntos que definen los hiperplanos mediante los que se generan las regiones para una clasificación En la figura 5.a) se presenta de forma gráfica una estructura general.

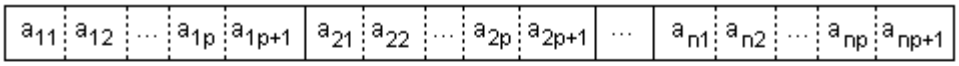

Fig.5.a) Representación de individuos COBLI

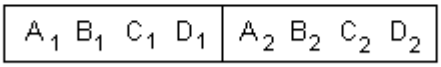

Fig.5.b) Ejemplo individuo COBLI

Donde, para una clasificación con n hiperplanos, que dividirá el espacio a clasificar en un máximo de 2^n regiones, y p atributos, la tupla $<a_{i1}, a_{i2},..., a_{ip},a_{ip+1}>$ representa los coeficientes que definen el hiperplano i. A modo de ejemplo, la figura 5.b) representa un individuo con dos hiperplanos, 2^2 regiones, en un espacio de tres atributos. De forma que, el hiperplano i queda definido por la ecuación $A_ix+B_iy+C_iz+D_i=0$

3.2. Algoritmo COBLI

El algoritmo principal de COBLI es un AG donde la población inicial se genera de manera aleatoria seleccionando entre los puntos pertenecientes al espacio a clasificar, incluyendo un número aleatorio de hiperplanos en cada individuo.

El proceso de evolución utiliza un AG elitista, el mejor individuo se mantiene para la nueva población. Un conjunto de individuos, seleccionados atendiendo a su bondad, pasarán a la siguiente generación, y el resto se obtendrán mediante cruces de los individuos existentes. El operador de cruce seleccionará dos individuos y generará uno nuevo compuesto por hiperplanos de los individuos originales seleccionados aleatoriamente.

El operador de mutación podrá afectar de dos formas a los individuos: desplazamiento o giro. El desplazamiento consiste en un movimiento paralelo del hiperplano mutado respecto a su posición original. Este efecto se consigue alterando una cantidad aleatoria y proporcional el término independiente del hiperplano. El giro consiste en la rotación del hiperplano mutado. El efecto del giro se consigue realizando una variación aleatoria y proporcional de algún/os coeficiente/s del hiperplano.

3.3. Función de bondad para COBLI

La función de bondad en COBLI determinará lo aceptable que es una clasificación realizada con los hiperplanos definidos en cada individuo. Para ello, se analizan cada uno de los hiperplanos determinando las regiones que forman, obteniendo una codificación de las regiones mediante una tira de 0 y 1. El valor en la posición i de la codificación indicará que la región se encuentra por encima (si es 1) o por debajo (si es 0) del hiperplano i del individuo. En la figura 6 se presenta la codificación de regiones para un ejemplo en dos dimensiones con dos hiperplanos, en este caso rectas.

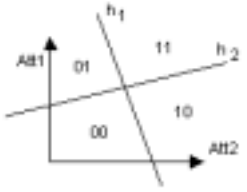

Fig. 6. Codificación de Regiones

Con esta codificación de regiones, el algoritmo analizará cada una de ellas estableciendo como *clase* para esa región aquella que tenga mayor número de elementos de dicha clase. Determinada la clase de una región, se calcula el porcentaje, *Clasp*, de elementos bien clasificados (3) y un factor de agrupación, A*grup*, que primará los individuos cuyas regiones mantengan un mayor número de elementos agrupados (4).

$$Clasp = \frac{\sum_{i=1}^{r} Elto\_Clas(i) - Elto\_NClas(i)}{r} \qquad (3)$$

$$Agrup = \frac{\sum_{i=1}^{r} \frac{(Elto\_Clas(i) - T\_Elto(i)) * FAGRUP}{T\_Elto(i)}}{r} \qquad (4)$$

Donde, para r regiones, Elto_Clas(i) es el número de elementos clasificados para la región i, Elto_NClas(i) es el número de elementos no clasificados en i, T_Elto(i) es el número total de elementos en la región i y FAGRUP es un factor de agrupación que establece un porcentaje por debajo del cual se penarán las clasificaciones.

La función de bondad definitiva, para cada individuo, se determina penando el número de regiones o hiperplanos, mediante el factor Nhip, de la clasificación según (5).

$$Bondad = Clasp + Agrup - Nhip * 0{,}01 - errors * 0{,}001 \qquad (5)$$

4. Parámetros utilizados en CGO y COBLI

Los parámetros son la piedra angular de un AG, y por ello, a veces su determinación es trascendental para obtener los resultados esperados. En la tabla 1 se presentan los principales parámetros utilizados en CGO y COBLI.

Parámetro	Valor
Número de generaciones	50
Número de individuos	50
Número de replicas	15%
Máx. componentes de un individuo	20
Número máx atributos	2
Probabilidad de Mutación	80%
Porcentaje de Mutación	70%

Tabla 1. Parámetros utilizados en CGO y COBLI

5. Aplicación y comparación de resultados.

5.1. Ejemplo ilustrativo

Para ver de forma gráfica los resultados obtenidos por los diferentes algoritmos, la figura 7 muestra un ejemplo creado artificialmente, formado por dos atributos y tres clases.

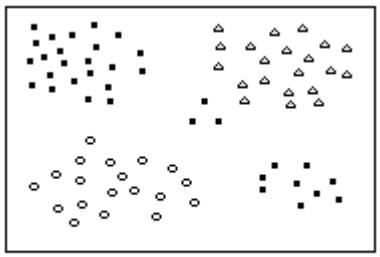

Fig.7. Base de Datos Artificial bd1

En la figura se aprecia claramente que una clasificación mediante un clasificador paralelo producirá un excesivo número de rectas. Sin embargo, con uno oblicuo se reduciría el número de rectas. En la figura 8.a. se presenta la solución obtenida por el C4.5. Por otra parte, no cualquier solución oblicua es buena. Si se utiliza el criterio de ganancia para colocar la primera recta, la solución obtenida podrá tener mas rectas de las necesarias. Este problema se produce debido a que el criterio de ganancia elegirá como mejor división aquella en la que se obtiene más información. En la figura 8.b. se presenta la solución alcanzada por el OC1, donde las líneas continuas muestran la solución simplificada o podada y el total de las rectas la solución no podada. Finalmente, los resultados obtenidos con CGO y COBLI se representan en las figuras 8.c. y 8.d., comprobándose la reducción del número de rectas en la clasificación.

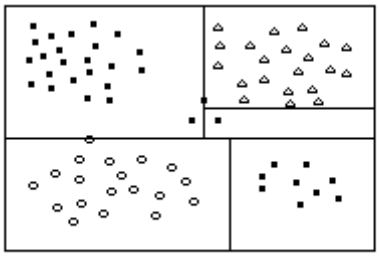

Fig.8.a) Clasificación obtenida con C4.5 para bd1

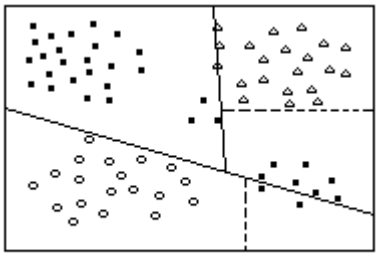

Fig.8.b) Clasificación obtenida con OC1 para bd1

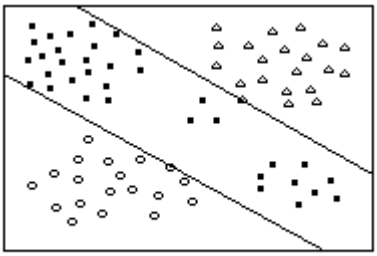

Fig.8.c) Clasificación obtenida con CGO para bd1

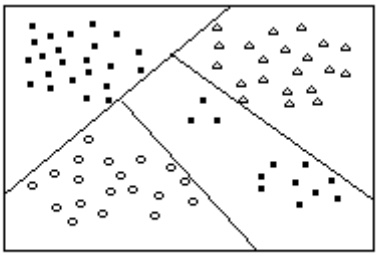

Fig.8.d) Clasificación obtenida con COBLI para bd1

5.2.- Tabla comparativa

A continuación se exponen una serie de resultados obtenidos con otras bases de datos artificiales compuestas por elementos generados aleatoriamente, distribuidos en regiones previamente definidas.

	C4.5		OC1		CGO		COBLI	
	error	rectas	error	rectas	error	rectas	error	rectas
bd1(100,2,3)	0%	4	0%	4	0%	2	0%	3
bd2 (200,2,3)	3,1%	10	0.5%	11	0.5%	9	1.5%	5
bd3 (200,2,2)	7,3%	18	1,1%	14	1,2%	8	0,5%	5

Tabla 2.- Resultados obtenidos con Bases de Datos Artificiales

6.- Referencias

[1] Quinlan, J.R. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, 1993.

[2] Murthy, S.K., Kasif, S. y Salzberg, S. A System for Induction of Obliques Decision Trees. *Journal of Artificial Intelligence Research*, 2, 1994, 1-32.
[3] Riquelme, J y Toro M. Metodología para obtener nuevas características en sistemas de aprendizaje. *CAEPIA'95*, 1995, 13-23.
[4] Alden H. Wright. Genetic Algorithm for Real Parameter Optimization. En: Rawlins J.G. *Foundations of Genetic Algorithms*. Morgan Kaufmann Publishers, 1991.
[5] Goldberg, D. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Pub. Company, inc. 1989.
[6] Aguilar, J., Riquelme, J. y Toro, M. Decision Queue Classifier for Supervised Learning Using Rotated Hyperboxes. Progress in Artificial Intelligence. Lecture Notes Artificial Intelligence 1484, 1998, 326-336