

LEX: A NEW ALGORITHM FOR THE CALCULUS OF ALL TYPICAL TESTORS

Yovanis Santiesteban Alganza, Aurora Pons Porrata
Dpto. de Computación, Universidad de Oriente, Santiago de Cuba, Cuba
{yosva | aurora}@csd.uo.edu.cu

ABSTRACT

The determination of all typical testors of a training matrix has great applications in Pattern Recognition, because they allow determining feature relevance in a classification problem and they are useful in the problems of feature selection. The determination of the typical testors has an exponential growth with respect to the number of features in the matrix. Therefore it is very important to find better algorithms for obtaining them. Methodologies that speed up the calculation of the set of all typical testors have been developed, but nowadays, there are still problems where the set of all typical testors is impossible to find. An external scale algorithm to calculate all typical testors, that is more efficient than any traditional algorithm, is presented. In order to show the performance of the proposed algorithm, its runtime was compared with the previously known algorithms in this area of Pattern Recognition.

Keywords: typical testor, algorithms to calculate all typical testors, feature relevance, Logical Combinatorial Pattern Recognition.

LEX: UN NUEVO ALGORITMO PARA EL CÁLCULO DE LOS TESTORES TÍPICOS

Yovanis Santiesteban Alganza, Aurora Pons Porrata
Dpto. de Computación, Universidad de Oriente, Santiago de Cuba, Cuba
{yosva | aurora}@csd.uo.edu.cu

RESUMEN

La determinación de todos los testores típicos de una matriz de aprendizaje encuentra gran aplicación en el Reconocimiento de Patrones, porque ellos permiten determinar el peso informacional de los rasgos e intervienen en los problemas de selección de variables. El tiempo requerido para la determinación de los testores típicos aumenta en forma exponencial con relación al número de rasgos que describen el problema. Por eso es de suma importancia el continuo intento por encontrar algoritmos de mejor desempeño en la obtención de los mismos. Se han desarrollado metodologías que aceleran el cálculo del conjunto de testores típicos, pero en la actualidad, existen aún problemas, donde el conjunto de testores típicos sigue sin poderse hallar. En el trabajo se propone un algoritmo de escala exterior orientado a tales propósitos que obtiene buenos resultados en su desempeño. Esto se demuestra con las experimentaciones realizadas para el algoritmo propuesto y los existentes en esta área del Reconocimiento de Patrones.

1. INTRODUCCIÓN

Uno de los problemas del Reconocimiento de Patrones (R.P.) es la selección de variables, que se utiliza para reducir de modo eficiente el número de variables (atributos o rasgos) con los cuales se deben describir los objetos y para encontrar los rasgos que inciden de manera determinante en un problema. En el Reconocimiento Lógico Combinatorio de Patrones [Shul95] una alternativa de solución al problema de la selección de variables es a partir de la utilización del conjunto de testores típicos.

El concepto de testor aparece por primera vez en un trabajo de los científicos rusos Chegus y Yablonskii [Cheg55] en un intento por tratar de simplificar la tarea de encontrar desperfectos en circuitos eléctricos que podían ser modelados a través de funciones del álgebra de la lógica. En 1966 un grupo de científicos rusos, al mando de Yu. I. Zhuravlev, dan un nuevo enfoque al concepto de testor [Dmit66] que permite aplicaciones importantes vinculadas a los problemas de clasificación con aprendizaje y selección de variables. En esencia, un testor es un conjunto de características (rasgos) que diferencia a elementos (objetos) de clases distintas. Un testor típico es un testor al que si le eliminamos cualquiera de sus rasgos pierde la propiedad de ser testor. Posteriormente este concepto ha sido extendido y variado para ajustarlo a otras aplicaciones. Ejemplos de estas extensiones son: los testores de Aizenberg y Tsipkin [Aize71] que consideran los rasgos asociados a dos predicados diferentes, los testores difusos de Goldman [Gold80] que se generan utilizando criterios de comparación reales, los testores de Andreev [Andr81] que trabajan

con clases no disjuntas, los k-testores primos de Shulcloper y Lazo [Shul91] que constituyen una extensión de los testores a la lógica k-valente, entre otros.

Los testores típicos son determinantes en problemas como la evaluación de la importancia informacional de los rasgos y la selección de variables, pues permiten reducir la dimensión del espacio de representación de los objetos y como sistemas de conjuntos de apoyo en múltiples algoritmos de clasificación, como por ejemplo ALVOT y KORA- Ω [Shul95]. Algunos trabajos donde se usan los testores típicos para evaluar la importancia informacional de los rasgos son: [Alba98], [Lazo99], entre otros.

En este trabajo se expone un nuevo algoritmo para el cálculo de todos los *testores típicos* de una matriz de aprendizaje. Cuando hablemos de testores típicos nos restringiremos aquí, a los testores típicos en el sentido clásico de Zhuravlev, es decir, aquellos surgidos de problemas del Reconocimiento de Patrones, donde las clases son conjuntos “duros” y disjuntos, los criterios de comparación por rasgos son booleanos y la función de semejanza entre objetos es la igualdad simple.

El artículo está estructurado como sigue. En la sección 2 se dan algunos conceptos básicos relacionados con los testores típicos para una mayor comprensión de la temática que se expone. La sección 3 contiene un breve estado del arte sobre los algoritmos existentes más representativos para el cálculo de testores típicos. El nuevo algoritmo propuesto se explica en la sección 4 y en la 5, se hace un análisis de las experimentaciones realizadas para mostrar el desempeño de este algoritmo. Finalmente, en la sección 6 se dan las conclusiones de nuestro trabajo.

2. CONCEPTOS PRELIMINARES

Sea $\Omega = \{O_1, O_2, \dots, O_m\}$ un conjunto de m objetos e $I(O_1), I(O_2), \dots, I(O_m)$ sus descripciones en términos de un conjunto de n rasgos o características $\mathfrak{R} = \{X_1, X_2, \dots, X_n\}$, donde cada rasgo X_j tiene asociado un conjunto M_j que se denomina *conjunto de valores admisibles del rasgo X_j* , es decir, $I(O_i) = (X_1(O_i), X_2(O_i), \dots, X_n(O_i))$, $X_j(O_i) \in M_j, j = 1, \dots, n, i = 1, \dots, m$. El tipo de una variable depende de la naturaleza de su conjunto de valores admisibles. Los conjuntos M_j pueden ser $\{0,1\}$, el conjunto de los números reales o de los enteros, $\{v_1, \dots, v_k\}$, donde los v_i no son necesariamente números, etc.

Se llama *criterio de comparación de valores de la variable X_j* a una función $C_j: M_j \times M_j \rightarrow L_j$, tal que si C_j es un *criterio de disimilaridad* se cumple que $C_j(X_j(O), X_j(O)) = \min_{y \in L_j} \{y\}$ y si C_j es un *criterio de similaridad* se cumple que $C_j(X_j(O), X_j(O)) = \max_{y \in L_j} \{y\}$, donde L_j es un conjunto totalmente ordenado, $j = 1, \dots, n$ [Shul95].

Ejemplo #1: Veamos ejemplos de criterios de comparación por rasgos (de disimilaridad):

1. $C_k(X_k(O_i), X_k(O_j)) = \begin{cases} 0 & \text{si } X_k(O_i) = X_k(O_j) \\ 1 & \text{e.o.c} \end{cases}$, llamado “criterio de comparación de igualdad estricta”

$$2. C_k(X_k(O_i), X_k(O_j)) = \begin{cases} 0 & \text{si } |X_k(O_i) - X_k(O_j)| \leq \varepsilon \\ 1 & \text{e.o.c.} \end{cases}, \text{ llamado "criterio de comparación de error admisible"}$$

El primero de los criterios es aplicable a rasgos definidos en cualquier dominio y el segundo sólo es aplicable a rasgos numéricos. En este último, ε es un valor real positivo.

Sean K_1, \dots, K_c un cubrimiento finito de subconjuntos propios de Ω . Se llama *c-uplo de pertenencia de un objeto O a las clases K_1, \dots, K_c* , denotado por $\bar{\alpha}(O)$, al tuplo $(\bar{\alpha}_1(O), \dots, \bar{\alpha}_c(O))$ donde $\bar{\alpha}_i(O) = \begin{cases} 1 & \text{si } O \in K_i \\ 0 & \text{si } O \notin K_i \end{cases}$ [Shul95]. Se le llama *Matriz de Aprendizaje (MA)* al tuplo $(I(O_1), \bar{\alpha}(O_1), I(O_2), \bar{\alpha}(O_2), \dots, I(O_m), \bar{\alpha}(O_m))$ que contiene las descripciones de los objetos conjuntamente con sus *c-uplos de pertenencia* [Shul95].

Ejemplo #2: Dada la siguiente información acerca de los rasgos en términos de los cuales se describen a los objetos:

	Tipo	Dominio de definición	Criterio de comparación	Datos adicionales
X_1	Continuo	$(-5.5, 5.5)$	error admisible	$\varepsilon = 0.5$
X_2	Discreto	$[10, 20]$	error admisible	$\varepsilon = 2.0$
X_3	Booleano	$\{\text{True}, \text{False}\}$	igualdad estricta	
X_4	<i>K</i> -valente	$\{\text{azul}, \text{blanco}, \text{rojo}, \text{verde}\}$	igualdad estricta	

una MA es la siguiente:

	X_1	X_2	X_3	X_4	$\bar{\alpha}$
O_1	1.3	10	True	Azul	1 0 0
O_2	-1.3	13	True	Rojo	1 0 0
O_3	2.4	16	False	Azul	0 1 0
O_4	-2.4	19	False	Blanco	0 1 0
O_5	2.6	10	True	Verde	0 0 1
O_6	-2.6	20	False	Rojo	0 0 1

Dada una MA y C_1, \dots, C_n criterios de comparación booleanos de cada uno de los rasgos, llamaremos *Matriz de Diferencias* de MA, y la denotaremos por MD, a la matriz booleana formada por las m' filas: $S_{ij} = (\delta_1^{ij}, \dots, \delta_n^{ij})$, $i, j = 1, \dots, m$, $i \neq j$, donde $\delta_p^{ij} = C_p(X_p(O_i), X_p(O_j))$, $p = 1, \dots, n$ y

$$m' = \sum_{i=1}^{c-1} \sum_{t=i+1}^c |K_i| |K_t| \text{ [Shul95].}$$

Sean i_p, i_q filas de MD. Diremos que i_p es *subfila* de i_q si $\forall(j)[a_{i_q j} = 0 \Rightarrow a_{i_p j} = 0]$ y además $\exists(j_0)[a_{i_q j_0} = 1 \wedge a_{i_p j_0} = 0]$. También diremos que i_q es *superfila* de i_p [Shul95a].

Sea i_q una fila de MD. La fila i_q es *básica* si no existe fila alguna i_p que sea subfila de i_q .

Dada una matriz de diferencias MD llamaremos *Matriz Básica* (MB) a la matriz formada exclusivamente por las filas básicas de MD [Shul95].

Ejemplo #3: Las matrices de diferencia y básica correspondientes a la MA del ejemplo 2 son las que se muestran a continuación:

MD:		MB:
<u>Filas</u>	<u>Objetos comparados</u>	
1110	O_1 con O_3	1110
1111	O_1 con O_4	0001
1001	O_1 con O_5	
1111	O_1 con O_6	
1111	O_2 con O_3	
1111	O_2 con O_4	
1101	O_2 con O_5	
1110	O_2 con O_6	
0111	O_3 con O_5	
1101	O_3 con O_6	
1111	O_4 con O_5	
0001	O_4 con O_6	

El subconjunto $\tau = \{X_{i_1}, \dots, X_{i_s}\}$ de rasgos de una MA es un *testor* si al eliminar de su MB todas las columnas, excepto las correspondientes a los elementos de τ , no existe fila alguna completa de ceros. τ constituye un *testor típico* (TT) si al quitarle cualquiera de sus rasgos deja de ser testor [Shul95]. Dicho de otro modo, siendo A la matriz formada sólo por las columnas correspondientes a los elementos de τ en la MB, τ es un testor típico si al eliminar cualquier columna de A aparece al menos una fila de ceros.

Ejemplo #4: Dada la matriz básica del ejemplo 3, son testores típicos los subconjuntos de rasgos $\{X_1, X_4\}$, $\{X_2, X_4\}$ y $\{X_3, X_4\}$. Sólo estos tres conjuntos cumplen esa propiedad para la matriz básica dada. Note que cualquier conjunto que contenga a un testor típico seguirá siendo testor, pero no típico.

3. ALGUNOS ALGORITMOS EXISTENTES

Para el cálculo de los testores típicos se han desarrollado varios algoritmos que por su estrategia de cómputo pueden clasificarse en algoritmos de escala exterior y de escala interior. Los algoritmos de escala exterior son aquellos que realizan el cálculo de los TT generando los

elementos del conjunto potencia del conjunto de columnas de la MB en un determinado orden (usualmente utilizando un n -uplo booleano característico [Shul95]), de tal forma que, usando determinados criterios, el algoritmo trata de evitar el análisis de todos los subconjuntos. Por el contrario, los algoritmos de escala interior realizan el cálculo de los TT basados en el estudio de la estructura interna de la matriz, encontrando condiciones que garantizan la característica de testor y la tipicidad en las columnas asociadas a las posiciones unitarias de la matriz.

Ejemplos de algoritmos de escala interior son CC [Agui84] y CT [Brav83] y de escala exterior BT [Shul82], TB [Shul82], REC [Shul95a] y CER [Ayaq97].

3.1. ALGORITMO BT

El algoritmo BT es de escala exterior [Shul82]. El orden que caracteriza a este algoritmo es el generado por los números naturales de forma ascendente en su notación binaria mediante un n -uplo booleano. Éste constituye un orden total sobre los elementos no nulos del conjunto potencia del conjunto de rasgos de la MA.

El BT comienza entonces por el n -uplo (0...01) y procede comprobando si cada vector (n -uplo) generado es un testor o un testor típico. Según el resultado establece “saltos” en el conjunto potencia. El algoritmo termina cuando llega al n -uplo (1...11). Para ello se apoya en caracterizaciones de los conceptos de testor y testor típico y en proposiciones que definen los saltos cuando el n -uplo analizado es o no un testor [Shul95].

En [Sanc97] se le hicieron modificaciones al algoritmo para mejorar su comportamiento. Estas modificaciones consisten en realizar un ordenamiento previo de la MB de modo que quede lo más parecido posible a una matriz triangular inferior para provocar la mayor cantidad de saltos posibles en el algoritmo y, por tanto, aumentar su eficiencia. Con este reordenamiento se logran dos condiciones importantes: la primera fila garantiza el mayor salto posible al comienzo del algoritmo y la primera fila que impide que se satisfaga la condición de testor provoca el mayor salto entre todas las que cumplen esta condición.

A nuestro juicio, el BT, incluso con la modificación, presenta dos inconvenientes. El primero consiste en que genera muchos n -uplos que son supraconjuntos de testores típicos. Esto es inherente al orden implantado y por tanto, los saltos que tiene definido sólo pueden evitar los supraconjuntos consecutivos al que acaba de detectar como testor y muchos de los supraconjuntos restantes serán examinados innecesariamente. El segundo de los inconvenientes está en que para comprobar si un nuevo n -uplo es o no testor siempre es preciso analizar las filas de la MB desde el principio, como si nunca antes se hubiese hecho.

3.2. ALGORITMO CT

El algoritmo CT [Brav83] es de escala interior y trabaja basado en los conceptos de *conjunto de filas independientes* y *conjunto completo*. Inicialmente se encuentran los testores y luego, se prueba si son típicos o no. Por definición todo conjunto completo es un testor. El método del CT consiste en explorar las posiciones unitarias de la MB encontrando los conjuntos completos y comprobando, entonces, si son TT.

Para mejorar el funcionamiento del algoritmo, se le aplicó un ordenamiento previo a la MB [Sanc97]. Este cambio, en esencia, consiste en colocar como primera fila de la MB aquella que tenga la menor cantidad de valores unitarios entre todas las filas de MB y si existe más de una fila

que cumpla esta propiedad, escoger aquella que tenga la mayor entropía global; o sea, aquella en la que la suma de los unos presentes en las columnas donde esta fila tiene valores unitarios sea mayor.

El CT, como los demás algoritmos de escala interior, presenta la deficiencia de que repiten los testores típicos encontrados. Este problema, a nuestro juicio, es inherente a su estrategia. Para que un conjunto τ sea un TT debe existir para cada rasgo de τ al menos una fila que tenga un 1 en esa columna y cero en todas las restantes de τ , pero esta fila no tiene por qué ser única; de hecho, es más probable que no lo sea. Esto se traduce para el CT en que un mismo TT se origina a partir de más de un par conjunto completo-conjunto de filas independientes.

3.3. ALGORITMO CC

El algoritmo CC [Agui84] es de escala interior y está basado en los conceptos de *Conjunto Compatible* y *Conjunto Regular*. Dos elementos unitarios de MB $a_{i_1j_1}$, $a_{i_2j_2}$ son compatibles si $a_{i_1j_2} = a_{i_2j_1} = 0$. El conjunto $D = \{ a_{i_1j_1}, \dots, a_{i_sj_s} \}$ de valores unitarios de MB es compatible si $s = 1$ ó sus elementos son compatibles dos a dos. Un conjunto regular es un conjunto compatible, en el cual las columnas de MB asociadas a él constituyen un testor.

El CC tiene como peculiaridad, a diferencia del CT, que su objetivo es garantizar la tipicidad. Esto está implícito en la definición de conjunto compatible, luego tiene que comprobar si se cumple la propiedad de testor.

El CC presenta la misma limitación señalada anteriormente al algoritmo CT, pues en este caso, un mismo TT se origina a partir de más de un conjunto compatible. Además el algoritmo tiene otra deficiencia: para obtener un TT es necesario explorar un gran número de combinaciones de las posiciones unitarias de la MB, es decir, de conjuntos compatibles.

3.4. ALGORITMOS REC Y CER

Ambos son algoritmos de escala exterior. El REC [Shul95a] tiene la particularidad de que trabaja directamente con la matriz de aprendizaje, lo cual lo sitúa en desventaja con relación a los restantes algoritmos por el enorme manejo de información que tiene que realizar. El CER [Ayaq97] fue encaminado a resolver este problema y lo hizo introduciendo un nuevo orden en el conjunto potencia de los rasgos. En la siguiente sección mostraremos los ordenamientos de los algoritmos de escala exterior tratados en el artículo, ya que lo consideramos un elemento primordial en el desempeño de cada algoritmo.

A pesar de las modificaciones que se le han realizado a los algoritmos existentes aún persiste el problema del incremento exponencial del tiempo necesario para el cálculo de los TT a medida que aumentan las dimensiones de las matrices de aprendizaje. En este campo se han desarrollado otras metodologías tales como la paralelización de algunos de estos algoritmos [Sanc97], una herramienta basada en algoritmos genéticos para calcular no todos los testores típicos, sino sólo los de costo mínimo [Sanc99], entre otras. El algoritmo que a continuación mostramos es un modesto intento de disminuir el tiempo necesario para la obtención de todos los TT de una matriz de aprendizaje.

4. ALGORITMO LEX

El algoritmo propuesto puede clasificarse como de escala exterior, debido a que la principal peculiaridad de este tipo de algoritmos es que buscan los TT implantando un orden sobre el conjunto potencia de los rasgos y definen ciertos saltos con el objetivo de obviar algunos conjuntos. Veamos a continuación el orden definido por este nuevo algoritmo.

Definamos la notación $[X|Y]$ para representar una lista ordenada de rasgos, de forma que X es el primer rasgo de la lista y Y es la lista ordenada de los restantes elementos. Note que es relevante el orden de los rasgos dentro de la lista. Así, por ejemplo, si tenemos la lista $[X_2, X_5, X_6]$ según esta notación podemos expresarlo como $[X_2|[X_5, X_6]]$ y, a su vez, $[X_5, X_6]$ como $[X_5|[X_6]]$.

« es una *relación de orden* sobre el conjunto potencia de \mathfrak{R} ($P(\mathfrak{R})$), tal que:

- Sean $Y = [X_i|Y']$ y $Z = [X_j|Z']$, entonces $Y \ll Z$ si $i < j$.
- Sean $Y = [X_i|Y']$ y $Z = [X_i|Z']$ entonces $Y \ll Z$ si $Y' \ll Z'$.
- Para toda Y se cumple que $[] \ll Y$.

A este orden le debe el algoritmo el nombre de LEX, pues es similar al orden lexicográfico en que son comparadas las cadenas de caracteres.

En la tabla #1 se muestra el ordenamiento sobre $P(\mathfrak{R})$ según los distintos algoritmos de escala exterior. Para simplificar se han representado los rasgos por sus índices en \mathfrak{R} y su representación binaria. Note las diferencias existentes entre cada uno de esos ordenamientos.

Tabla #1:

REC $X \subseteq \mathfrak{R}$ Bin.		CER $X \subseteq \mathfrak{R}$ Bin.		BT $X \subseteq \mathfrak{R}$ Bin.		LEX $X \subseteq \mathfrak{R}$ Bin.	
123	111	\emptyset	000	\emptyset	000	\emptyset	000
12	110	1	100	3	001	1	100
13	101	2	010	2	010	12	110
1	100	3	001	23	011	123	111
23	011	12	110	1	100	13	101
2	010	13	101	13	101	2	010
3	001	23	011	12	110	23	011
\emptyset	000	123	111	123	111	3	001

En lo adelante, cuando hablemos de lista nos estaremos refiriendo a una lista ordenada de rasgos.

Con el operador $+$ definimos la concatenación de listas. Así: $[X_1, X_2] + [X_3, X_4] = [X_1, X_2, X_3, X_4]$.

Sea una lista de rasgos $l = [X_{j_1}, \dots, X_{j_s}]$. Denominaremos *conjunto de filas típicas de X_t con respecto a l* al conjunto de los índices de las filas de MB que tienen valores unitarios en la columna correspondiente a X_t y ceros en las correspondientes a X_{j_k} , $1 \leq k \leq s$, $X_{j_k} \neq X_t$ y lo denotaremos por $F_{X_t}^l$. Note que X_t no necesariamente debe pertenecer a l .

Ejemplo #5: Dada la MB siguiente:

001100
100000
000101
000010
010001

y la lista $l=[X_1, X_3]$ entonces $F_{X_6}^l = \{3,5\}$

Denotaremos por cr_i^l a la cantidad de rasgos de l que tienen valores unitarios en la i -ésima fila de MB.

Proposición 1: Sean $l = [X_{j_1}, \dots, X_{j_s}]$ y un rasgo cualquiera $X_t \notin l$. Si existe una fila i de MB tal que cumple:

- 1) $MB_{it} = 1$.
- 2) $cr_i^l = 1$ siendo X_{j_k} el rasgo de l con valor unitario en la fila i , $1 \leq k \leq s$.
- 3) $\left| F_{X_{j_k}}^l \right| = 1$.

Entonces $F_{X_{j_k}}^l = \{i\}$ y X_t no formará parte de ningún TT junto con todos los rasgos de l .

Demostración:

De 2 tenemos que $i \in F_{X_{j_k}}^l$ por definición de filas típicas.

Entonces de 3 se deriva que $F_{X_{j_k}}^l = \{i\}$.

Supongamos que $\tau = l + [X_t] + f$ es un testor típico, donde f es una *lista ordenada de rasgos* que no tiene elementos en común con l ni con $[X_t]$. Entonces, por definición de testor típico, si eliminamos, por ejemplo, el rasgo X_{j_k} , $1 \leq k \leq s$ de τ , debe aparecer una fila p de MB completa de ceros en la matriz formada por las columnas correspondientes a los rasgos de τ , exceptuando a X_{j_k} . Por tanto, $p \in F_{X_{j_k}}^l$ y por 3 se cumple que esta fila p es única y tiene que ser i . Esto significa que ningún otro rasgo de τ diferente a X_{j_k} puede tener valor unitario en la fila i , pero esto se contradice con 1. Por tanto, τ no puede ser un testor típico.

l.q.q.d.

Proposición 2: Sean $l = [X_{j_1}, \dots, X_{j_s}]$ una lista no vacía de rasgos de MB y un rasgo cualquiera $X_t \notin l$. Si $\left| F_{X_t}^l \right| = 0$, entonces X_t no formará parte de ningún TT junto con todos los rasgos de l .

Demostración:

Supongamos que $\tau = l + [X_t] + f$ es un testor típico, donde f es una *lista ordenada de rasgos* que no tiene elementos en común con l ni con $[X_t]$. Entonces, por definición de testor típico, debe existir al menos una fila i de MB que tenga un uno en la columna correspondiente a X_t y cero en todas las columnas correspondientes a los rasgos de $l + f$. Por tanto, por definición, i es una fila típica de X_t con respecto a $l + f$ y, en particular, i es una fila típica de X_t con respecto a l . Esto contradice el hecho de que $\left| F_{X_t}^l \right| = 0$.

l.q.q.d.

Si se cumple al menos una de las dos proposiciones anteriores diremos que X_t es *excluyente con* l .

Ejemplo #6: Dada la MB y la lista del ejemplo anterior tenemos que X_4 es excluyente con dicha lista, pues cumple la proposición 1. Note que, en este caso, no se cumple la proposición 2.

Sea $l = [X_{j_1}, \dots, X_{j_s}]$ una lista de rasgos. Denominaremos *hueco* de l al rasgo $X_p \in l$, $j_1 \leq p \leq j_s$, tal que $p = \max \{ j_q / X_{j_q}, X_{j_{q+1}} \in l \wedge j_{q+1} \neq j_q + 1 \}$, es decir, es el mayor índice de los elementos de l tal que el índice de su consecutivo en l no es su consecutivo en MB.

Proposición 3: Sea $l = [X_{j_1}, \dots, X_{j_s}]$ una lista asociada a un testor típico tal que $X_{j_s} = X_n$. Si existe X_p hueco de l , sea $l' = [X_{j_1}, \dots, X_{j_k}, X_{p+1}]$, donde $j_1 \leq \dots \leq j_k = p-1 < p < j_s$. Entonces no existe ninguna lista λ tal que $l \ll \lambda \ll l'$ y λ esté asociada a un testor típico.

Demostración:

Denotemos l como $\eta + \delta$, donde $\delta = [X_{j_q}, X_{j_{q+1}}, \dots, X_{j_s}]$ es el sufijo de l , $j_q = p+1$ y $\eta = [X_{j_1}, \dots, X_{j_k}, X_p]$ es el prefijo de l . Entonces, por el orden definido, toda lista $l \ll \lambda \ll l'$ podemos denotarla como $\lambda = \eta + v$, donde v es el sufijo de λ . Como todos los rasgos de δ son consecutivos en MB y $X_{j_s} = X_n$, entonces todo rasgo de v está en el sufijo de l . Esto implica que cualquiera sea v , el conjunto de rasgos representado por v es subconjunto del conjunto de rasgos representado por δ . Por tanto, todo subconjunto de rasgos representado por λ será subconjunto del subconjunto de rasgos τ representado por l y como τ es un testor típico ninguna lista λ podrá estar asociada a un testor típico.

l.q.q.d.

Corolario 1: Sea $l = [X_{j_1}, \dots, X_{j_s}]$ una lista asociada a un testor típico tal que $X_{j_s} = X_n$. Si no existe hueco de l , entonces no existe ninguna lista λ tal que $l \ll \lambda$ y λ esté asociada a un testor típico.

Corolario 2: Sea $l + [X]$ una lista asociada a un no testor tal que $X = X_n$ y $l = [X_{j_1}, \dots, X_{j_s}]$. Si existe X_p hueco de l , sea $l' = [X_{j_1}, \dots, X_{j_k}, X_{p+1}]$, donde $j_1 \leq \dots \leq j_k = p - 1 < p < j_s$. Entonces no existe ninguna lista λ tal que $l \ll \lambda \ll l'$ y λ esté asociada a un testor.

4.1. EXPLICACIÓN GENERAL

El algoritmo LEX va generando listas de rasgos siguiendo el orden explicado en la sección anterior. La idea del algoritmo es ir construyendo listas de rasgos que posean la propiedad de tipicidad y luego, comprobar si este conjunto de rasgos constituye un testor. Cuando se obtiene un testor típico se producen saltos.

El algoritmo comienza analizando el primer rasgo de MB. Un nuevo rasgo se puede incorporar a la lista si se cumple que no es excluyente con la lista (puede coexistir con los rasgos de la lista para formar un testor típico y tiene filas típicas con respecto a la lista). Esta condición es la que garantiza la propiedad de tipicidad.

Luego, se comprueba si el conjunto de rasgos contenido en la lista junto con el nuevo rasgo forma un testor, verificando si no existe fila en la MB (considerando sólo esos rasgos) completa de ceros. Si se cumple que es testor, entonces estamos en presencia de un testor típico y se almacena.

Si se encontró un testor típico y éste contiene al último rasgo de MB, entonces se saltan todos sus subconjuntos consecutivos. Si no contiene al último rasgo de MB, para saltar todos los supraconjuntos del testor típico encontrado se elimina el último rasgo de la lista y se analiza si se puede incluir el próximo rasgo de MB.

Cuando un rasgo es aceptado como nueva componente de la lista actual l se actualizan adecuadamente los cr_i^l para todas las filas de MB y los conjuntos de filas típicas de todos los rasgos de l .

Si el rasgo analizado no se puede incorporar a la lista se prosigue el análisis con el siguiente rasgo de la matriz básica.

Con el objetivo de lograr un mejor desempeño del algoritmo se realiza previamente un ordenamiento de MB que consiste en colocar como primera fila aquella que tenga la menor cantidad de unos y, en esta fila encontrada, situar todos los unos a la izquierda (lo cual implica, por supuesto, un reordenamiento de las columnas correspondientes a cada posición unitaria). Si existiera más de una fila con mínima cantidad de unos se toma cualquiera de ellas. De esta forma, el algoritmo puede terminar cuando, durante el proceso de retroceso, la lista actual esté vacía y el rasgo que le corresponda analizarse posea un cero en la primera fila, ya que ninguno de los posibles subconjuntos de rasgos que se pueden formar con los rasgos que restan pueden formar un testor típico por no cumplir la propiedad de ser testor (la primera fila de la MB es completa de ceros en estas columnas).

4.2. DESCRIPCIÓN DEL ALGORITMO

Entrada: Matriz básica MB

Salida: El conjunto de todos los testores típicos de MB

1. Ordenación de MB

- a. Encontrar la fila con cantidad mínima de unos; de existir más de una, escoger cualquiera de ellas. Ponerla como primera fila en MB.
- b. Ordenar las columnas poniendo indistintamente como primeras las que tengan un 1 en la primera fila.

2. Inicialización

- a. $l = []$, $X = X_1$ (X es el primer candidato a elemento de l)

3. Evaluación del candidato X

- a. Si $l = []$ y la columna correspondiente a X tiene cero en la primera fila de MB entonces FIN.
- b. Si X es excluyente con l entonces ir al paso 4; no se acepta el candidato (proposición 1 y 2).
- c. Si para toda fila i de MB $cr_i^{l+[X]} > 0$, entonces Guardar $l + [X]$; es un TT. Ir al paso 4.
- d. Si $X = X_n$, entonces Ir al paso 4 (Corolario 2).
- e. Hacer $l = l + [X]$; se acepta el candidato. Actualizar los valores de cr_i^l para todas las filas de MB y $F_{X_i}^l$ para todos los elementos X_t en l .

4. Selección del nuevo candidato

- a. Si $X \neq X_n$, entonces sea j el índice de X en MB, hacer $X = X_{j+1}$ e ir al paso 3.
- b. Si $l = []$ entonces FIN.
- c. Si $l + [X]$ fue un TT o X no fue excluyente con l , entonces Si existe el hueco X_p de $l+[X]$, entonces hacer $X=X_{p+1}$ y eliminar de l todos los elementos desde X_p hasta X_{j_s} (proposición 3), actualizando cr_i^l para todas las filas de MB y $F_{X_i}^l$ para cada X_t de l . Ir al paso 3.
De no existir X_p entonces FIN (corolario 1).
- d. Hacer $X = X_{j_s}$, donde X_{j_s} es el último rasgo de l y $l=l \setminus [X_{j_s}]$. Actualizar cr_i^l para todas las filas de MB y $F_{X_i}^l$ para cada X_t de l . Retornar al paso 4.

5. EXPERIMENTACIÓN

Para comparar el desempeño de los algoritmos analizados y el propuesto se utilizaron matrices de diferencia generadas aleatoriamente de diferentes dimensiones. A pesar de que ellas no provienen de un problema práctico concreto cumplen los requerimientos según nuestro propósito.

En la tabla #2 se muestran algunas de las pruebas realizadas a los algoritmos, encaminadas a comparar el tiempo que consume cada uno en el cálculo de los TT para matrices de diversas dimensiones. Estas corridas fueron realizadas en una Pentium 2 de 300 MHz con 128 RAM.

Como puede observarse el algoritmo de escala exterior BT tiene mejor desempeño que los de escala interior (CC y CT), incluso para matrices de pequeña dimensión. Aún siendo lento, el CT fue más rápido que el CC en todas las experimentaciones realizadas. El CC para matrices de una dimensión superior a 55x20 es muy lento y, por tanto, podemos descartarlo como posible solución a un problema práctico.

Precisamente por lo anterior, se hizo énfasis en la confrontación del algoritmo expuesto con el BT, no sólo por la similitud de las técnicas usadas, sino porque es precisamente el BT el de mejor desempeño. Es por eso que en la tabla, en la mayoría de los casos, sólo aparecen los tiempos para el BT y el LEX.

En la tabla se muestra la diferencia significativa que existe en los tiempos de cálculo del BT y el LEX a medida que aumenta la dimensión de las matrices.

Tabla #2:

Orden de MB	CC	CT	BT	LEX	# de TT
22x15	14832 mseg.	571mseg.	30 mseg.	10 mseg.	165
24x16	39697 mseg.	2824 mseg.	70 mseg.	30 mseg.	230
29x17	3 min.	4457 mseg.	200 mseg.	50 mseg.	430
54x20	1h.	4 min.	2424 mseg.	481 mseg.	2294
59x25		2h.	47999 mseg.	2103 mseg.	11746
97x25			60033 mseg.	8172 mseg.	19973
58x30			10 min.	30 seg.	34900
65x35			Más de 1 h. 30 min	1 min.	112219
70x40			Más de 2 h. 53 min.	2 min.	446986
100x60			Más de 12 h.	2 h.	16567230

6. CONCLUSIONES

En este trabajo se propuso un nuevo algoritmo de escala exterior que tiene significativamente mejor desempeño que los algoritmos analizados. La tabla mostrada en la sección anterior ha sido bastante explícita y sólo ha incluido parte de todas las pruebas realizadas. En todas el resultado fue el mismo: el orden de los algoritmos según su desempeño en forma descendente fue LEX, BT, CT y, por último, el CC.

LEX eliminó los dos inconvenientes del algoritmo BT señalados en la sección 3.1. Precisamente por el orden de recorrido usado para el conjunto potencia de los rasgos, al encontrar un testor típico evita el análisis de todos los supraconjuntos que le suceden en el orden establecido, lo cual no se logra con el BT y evita también el análisis de todos los subconjuntos consecutivos en el orden establecido. Además la existencia y actualización para cada lista del conjunto de filas típicas de los

rasgos de dicha lista y de los cr_i^l para todas las filas de MB es muy útil para impedir la repetición de cálculos y poder reutilizarlos en el proceso de conformación de los próximos testores típicos.

El ordenamiento previo que se realiza de la matriz básica en el LEX requiere de mucho menos cálculos que el que se realiza en el BT. Además, en el caso del BT, cuando la MB es compleja (en cuanto a la disposición de ceros y unos en ella) puede suceder que a pesar de hacer el reordenamiento no se logre saltar prácticamente ningún n -uplo. Este caso sucede si no se logra conformar una “buena” matriz triangular. Sin embargo, en el LEX, independientemente de la disposición de ceros y unos en la MB, la primera fila después del ordenamiento siempre garantiza la generación de la menor cantidad de listas, lo cual constituye el objetivo trazado para dicho ordenamiento.

A nuestro juicio, estos tres elementos determinan el mejor comportamiento del algoritmo expuesto.

7. AGRADECIMIENTOS

Los autores quisieran agradecer los valiosos comentarios y sugerencias del Dr. Manuel Lazo Cortés del ICIMAF y del Dr. Guillermo Sánchez Díaz de México.

8. REFERENCIAS

- [Agui84] Águila, L.; Shulcloper, J.R. “Algoritmo CC para la elaboración de la información k-valente en problemas de Reconocimiento de Patrones”. *Revista Ciencias Matemáticas*, Vol. 5, No. 3, 1984.
- [Aize71] Aizenberg, N. N.; Tsipkin, A. I. “Testores primos”. *Doklady Akademii Nauk SSSR* 201, pág. 801-802, 1971 (en ruso).
- [Alba98] Alba Cabrera, E.; Lazo Cortés, M. “Una solución global para la utilización de los testores en problemas de reconocimiento de patrones”. *Memorias del III Taller Iberoamericano de Reconocimiento de Patrones*, Ed. IPN, México, pp. 209-217, 1998.
- [Andr81] Andreev, A. E. “On irredundant and minimal tests”, *Soviet Mathematics Doklady* Vol. 23, No. 1, pág. 92-96, 1981.
- [Ayaq97] Ayaquica, I.O. “Un nuevo algoritmo de escala exterior para el cálculo de testores típicos”. *Memorias del II Taller Iberoamericano de Reconocimiento de Patrones*, La Habana, pág. 141-148, 1997.
- [Brav83] Bravo, A. “Algoritmo CT para el cálculo de los testores típicos de una matriz k-valente”. *Revista Ciencias Matemáticas*, Vol. 4, No. 2, Cuba, pág. 123-144, 1983.
- [Cheg55] Chegus, I. A.; Yablonskii, S. V. “Acerca de los tests para esquemas eléctricos”. *Uspieji matematicheskij Nauk*, tom 4, #10, pp. 182-184, Moscú, 1955 (en ruso).
- [Dmit66] Dmitriev, A. N.; Zhuravlev, J. I.; Krendeleiev, F. P. “Acerca de los principios matemáticos de la clasificación de objetos y fenómenos”. *Diskretnyi Analiz* 7, pág. 3-15, 1966 (en ruso).
- [Gold80] Goldman, R. S. “Problemas de la teoría de los test difusos” (en ruso). *Automátika y Telemejánika*, No. 10, pág. 146-153, 1980.
- [Lazo99] Lazo Cortés, M.; Sánchez Díaz, G.; Quintana Gómez, T. “Evaluación de la relevancia de los rasgos en un problema de clasificación supervisada a partir de todos los testores”.

Memorias del IV Simposio Iberoamericano de Reconocimiento de Patrones SIARP'99, Ciudad Habana, pág. 215-222, 1999.

- [Sanc97] Sánchez Díaz, G. “*Desarrollo y programación de algoritmos eficientes (secuenciales y paralelos) para el cálculo de los testores típicos de una matriz básica*”. Tesis de Maestría en Ciencias de la Computación, BUAP, Puebla, México, 1997.
- [Sanc99] Sánchez Díaz, G.; Lazo Cortés, M.; Fuentes Chávez, O. “Algoritmo genético para calcular testores típicos de costo mínimo”. *Memorias del IV Simposio Iberoamericano de Reconocimiento de Patrones SIARP'99*, Ciudad Habana, pág. 207-213, 1999.
- [Shul82] Shulcloper, J.R.; Bravo, M.A.; Águila, F.L. “Algoritmos BT y TB para el cálculo de todos los tests típicos”. *Revista Ciencias Matemáticas*, Vol. 6, No. 2, 1982.
- [Shul91] Shulcloper, J. R.; Lazo Cortés, M. “K-testores primos”, *Ciencias Técnicas, Físicas y Matemáticas* 9 , pág.17-55, 1991.
- [Shul95] Shulcloper, J. R.; Alba Cabrera, E.; Lazo Cortés, M. “*Introducción al Reconocimiento de Patrones (Enfoque Lógico-Combinatorio)*”. Serie Verde No. 51, CINVESTAV-IPN, México, 1995.
- [Shul95a] Shulcloper, J. R.; Alba Cabrera, E.; Lazo Cortés, M. “*Introducción a la teoría de Testores Típicos*”. Serie verde 50, CINVESTAV-IPN, 1995.