Algortimo para el cálculo de todos los Testores Típicos de una matriz básica

Matemáticas Discretas (2627)

Daniel Martínez 00329519
Jarod Tierra 00331798
Gabriel Cazares 00331514
Andrés Vega 00331970

24 de noviembre de 2023

Eduardo Alba
Primer semestre 2022-2023

1. Introducción

En los últimos años, ha surgido una teoría innovadora que permite crear un subconjunto de características capaz de discriminar entre clases sin dejar de ser inclusivo. Esta aproximación ha encontrado aplicaciones que van desde el diagnóstico de enfermedades hasta la categorización de texto. Aunque existen varios algoritmos para calcular los testores típicos de una matriz binaria, este documento se centra en detallar el método YYC, una técnica novedosa y prometedora en este ámbito.

2. EJERCICIO 1

X6	X7	X8	X9	X10
1	1	1	0	0
1	1	0	0	1
1	0	1	1	0
1	0	1	0	1

TT1 = $\{\{X6\}, \{X7\}, \{X8\}\}$
Se aumenta la línea 2 al análisis,

La variable X6 y X7 siguen siendo testores, sin embargo la Variable X8 pierde la propiedad de Testor ya que aparece una fila de ceros, por lo tanto tenemos que

buscar si se puede añadir un 0.
Probamos con $\{X8, X6\}$

X6	X8
1	1
1	0

Comprobamos que cumple con la propiedad de testor, sin embargo, el número de filas que solo poseen un 1 no es $\geq$ que la cardinalidad de las variables por lo que ya no se cumple la propiedad de tipicidad y lo descartamos.

Probamos con $\{X8, X7\}$

X7	X8
1	1
1	0

Se cumple la propiedad de testor, sin embargo, el número de filas que solo poseen un 1 no es $\geq$ que la cardinalidad de variables y ya no se cumple la propiedad de tipicidad, por ello lo descartamos.

Probamos con $\{X8, X10\}$

X8	X10
1	0
0	1

Comprobamos que cumple como testor, que el número de filas es $\geq$ a la cardinalidad de las variables, y que la suma de las columnas da un número ≥ 1, se cumple la tipicidad y pasa como testor Típico.

TT2 = $\{\{X6\}, \{X7\}, \{X8, 10\}\}$

Se aumenta la línea 3 al análisis

Los testores $\{X6\}$ y $\{X8, 10\}$ se siguen manteniendo, sin embargo el testor $\{X7\}$ pierde la propiedad pues aparece una fila completa de ceros, por lo que debemos realizar el análisis para ver que 1 se podría llegar a agregar.

Probamos con $\{X7, X6\}$

X7	X6
1	1
1	1
0	1

Comprobamos que se cumple la propiedad de Testor, sin embargo, el número de filas que solo poseen un 1 no son $\geq$ que el número de Variables por lo que no se cumple tipicidad y se descarta.

Probamos con $\{X7, X8\}$

X7	X8
1	1
1	0
0	1

Cumple con ser Testor, el número de filas con solamente un 1 es igual al número de variables por lo que cumple la condición y cuando se suma cada columna nos da un número $\geq$ a 1, por lo que se cumple la propiedad de Tipicidad y se incluye el testor Típico.

Probamos con $\{X7, X9\}$

X7	X9
1	0
1	0
0	1

Se cumple que es Testor, el número de filas que poseen tan solo un 1 es mayor que la cardinalidad de las variables por lo que se cumple la condición 1 de Tipicidad y al sumar cada columna, el resultado es $\geq$ a 1 por lo que se incluye el Testor Típico.

TT3 = $\{\{X6\}, \{X7, X8\}, \{X7, X9\}, \{X8, 10\}\}$

Se aumenta la línea 4 al análisis.

Los testores $\{X6\}$, $\{X8, X10\}$ y $\{X7, X8\}$ se siguen manteniendo pero $\{X7, X9\}$ pierde su propiedad de ser Testor, por lo tanto tenemos que buscar si se le puede agregar un 1.

Probamos con $\{X7, X9, X6\}$

X7	X9	X6
1	0	1
1	0	1
0	1	1
0	0	1

Se cumple con la condición de Testor, sin embargo, cuando se cuenta las lineas que solo poseen un 1, resulta que son menos que la cardinalidad de las variables ya que 1<3, por lo tanto no lo tomamos en cuenta.

Probamos con $\{X7, X9, X8\}$

X7	X9	X8
1	0	1
1	0	0
0	1	1
0	0	1

Igualmente, se cumple la propiedad de Testor, pero nos damos cuenta que las líneas que solo poseen un 1 son menos que el número de Variables que se analizan ya que 2<3, por lo tanto ya no lo tomamos en cuenta.

Probamos con $\{X7, X9, X10\}$

X7	X9	X10
1	0	0
1	0	1
0	1	0
0	0	1

Al igual que los otros, se conserva la propiedad de Testor ya que no aparece una fila completa de ceros. Nos podemos dar cuenta que el número de filas que poseen tan solo un uno son 3 y dado que la cantidad de variables son 3; se cumple la condición que el número de filas sea $\geq$ que la cardinalidad de variables, Pasando a la condición 2, cuando se suma cada columna se obttiene siempre un número ≥ 1, por lo que se cumple la Tipicidad.

Terminado el Análisis, se puede concluir que los Testores Típicos son:

TT = $\{\{X6\}, \{X7, X8\}, \{X7, X9, X10\}, \{X8, 10\}\}$

3. EJERCICIO 2

Dentro del ejercicio 2 se realizó una implementación computacional del algoritmo YYC en lenguaje de Python. Este algoritmo realiza una serie de operaciones sobre una matriz dada para identificar y obtener el conjunto de testores típicos.

Primero, se definen varias funciones:

1. Verificar tipicidad fila

```
def verificar_tipicidad_fila(fila, matriz, testores, testor_ubi):
    suma = sum(matriz[fila][i] for i in testores[testor_ubi])
    return suma >= 1
```

La función tiene los siguientes argumentos:

- "fila": es el índice de la fila que se está verificando en la matriz.
- "matriz": es la matriz binaria en la que se está verificando la tipicidad.
- "testores": es una lista de testores.
- "testor_ubi": indica la lista que contiene los testores con los que se va a trabajar.

La función calcula la suma de los valores en la "fila"de la matriz, pero solo considera las columnas seleccionadas por el testor identificado por "testor_ubi". Esto se realiza usando una comprensión de lista donde se suman solo los elementos de la "fila"que corresponden a las columnas del testor. Finalmente, return suma ¿= 1 devuelve True si la suma es mayor o igual a 1, lo que significa que al menos una columna seleccionada por el testor contiene el valor 1 en la fila actual. De lo contrario, devuelve False. Esto se realiza con el fin de que si se tiene un conjunto de testores antes de someterlos a las demas pruebas verificar directamente si en la fila siguiente siguen siendo testores típicos.

2. Compatibilidad Testores:

```python
def compatibilidad_testores(col, testores, testor_ubi, fila, matriz):
    testores_tip = testores[testor_ubi][:]
    matriz_apoyo = [[matriz[i][indice] for indice in testores[testor_ubi]] +
                    [matriz[i][col]] for i in range(fila + 1)]
    cnt = sum(sum(fila) == 1 for fila in matriz_apoyo)

    if cnt >= len(testores[testor_ubi]):
        matriz_apoyo = [fila for fila in matriz_apoyo if sum(fila) <= 1]
        todos_mayores_o_iguales_a_uno = all(sum(columna) >= 1 for columna in zip(*matriz_apoyo))

        if todos_mayores_o_iguales_a_uno:
            testores_tip.append(col)
        else:
            testores_tip.clear()
    else:
        testores_tip.clear()

    return testores_tip
```

La función tiene las siguientes argumentos:

- "col": indica la columna que quiero agregar.
- "testor_ubi": es el índice que indica qué testor específico se está evaluando.
- "fila": es el índice de la fila actual en la matriz.
- "matriz": es la matriz binaria en la que se está evaluando la compatibilidad.
- "testores": es una lista de testores.

Testores tip:
Inicializa testores_tip como una copia del testor específico que está siendo evaluado.

Matriz apoyo:
Se construye una matriz la cual tiene los elementos que me interesan que en ese caso son los de la matriz original en la fila especificada y las columnas correspondientes a los tetsores tipicos en la ubicacion pasada como parametro, adicionalmente se le agrega la columna que quiero agregar. Por lo tanto, me queda una matriz unicamente con los elementos de interes.

Cnt:
Cuenta cuántas filas en "matriz_apoyo"tienen exactamente un elemento igual a 1.

La siguiente parte:

```python
if cnt >= len(testores[testor_ubi]):
```

Comprueba si el contador cnt es mayor o igual a la cardinalidad del conjunto de testores actual. Si es así, continúa con las siguientes operaciones.

Matriz apoyo :
Filtra matriz_apoyo manteniendo solo las filas donde la suma de elementos sea menor o igual a 1.

Todos mayores o iguales a uno:
Verifica si todos los elementos de las columnas seleccionadas suman un valor mayor o igual a 1.

Si se cumplen las condiciones anteriores, se agrega col a "testores_tip"si es compatible con el testor, de lo contrario, se limpia "testores_tip". Finalmente, se devuelve "testores_tip", que contiene el testor actualizado si es compatible con las condiciones especificadas, o vacío si no lo es.

3. Completar testores típicos

```
def completa_testores_tipicos(fila, matriz, testores):
    testores_nuevos = []
    n_columnas = len(matriz[0])

    for i in range(len(testores)):
        if not verificar_tipicidad_fila(fila, matriz, testores, i):
            columnas_a_probar = [j for j in range(n_columnas) if matriz[fila][j] == 1]
            for k in columnas_a_probar:
                testor_apoyo = compatibilidad_testores(k, testores, i, fila, matriz)
                if testor_apoyo:
                    testores_nuevos.append(testor_apoyo)
        else:
            testores_nuevos.append(testores[i])

    testores[:] = testores_nuevos
```

La función tiene los siguientes argumentos:

- fila: Índice de la fila actual que se está considerando en la matriz.
- matriz: La matriz binaria en la que se están evaluando los testores.
- testores: La lista de testores que se están actualizando.

Testores nuevos:
Inicializa una lista vacía para almacenar los nuevos testores típicos.

Columnas:
Obtiene la cantidad de columnas en la matriz.

Para cada testor existente en la lista testores:
Se verifica si la fila actual cumple con la tipicidad para el testor `i`. Si no cumple, se procede a encontrar nuevas columnas para probar. Se seleccionan las columnas de la fila actual que tienen el valor 1. Esto se realiza utilizando una comprensión de lista que itera sobre las columnas de la fila y selecciona aquellas que tienen el valor 1.

Para cada columna seleccionada:

Se verifica la compatibilidad del testor `i` con esa columna mediante la función `compatibilidad_testores`. Si el `testor_apoyo` es válido, se agrega a testores_nuevos.

Si la fila cumple con la tipicidad para el testor actual, se agrega directamente el testor existente testores[i] a testores_nuevos, y finalmente, testores[:] = testores_nuevos actualiza la lista de testores testores con la nueva lista testores_nuevos, sobrescribiendo así los testores anteriores con los testores actualizados que cumplen con las condiciones especificadas en la teoría de testores.

Para finalizar el programa, se realiza la parte principal:

Programa principal

```
# Programa principal
testores_tipicos = []
matriz = [
    [1, 0, 0, 0],
    [0, 1, 1, 0],
    [0, 1, 0, 1],
    [0, 0, 1, 1],

]

for fila in matriz:
    print(' '.join(map(str, fila)))

testores_tipicos = [[i] for i, val in enumerate(matriz[0]) if val == 1]

for i in range(1, len(matriz)):
    completa_testores_tipicos(i, matriz, testores_tipicos)

for testor in testores_tipicos:
    print("{ " + ' '.join(f"x{index + 1}" for index in testor) + " }")
```

Dentro del programa principal se encuentra lo siguiente:

1. testores_tipicos: Inicializa una lista vacía que almacenará los testores típicos encontrados en la matriz.
2. matriz : Define una matriz binaria de ejemplo.
3. Se imprime la matriz en la consola fila por fila usando un bucle for.
4. testores_tipicos: Inicializa testores_tipicos con los índices de las columnas que contienen el valor 1 en la primera fila de la matriz.
5. Luego, se ejecuta un bucle `for` desde la fila 1 hasta el final de la matriz (excluyendo la primera fila) utilizando `completa_testores_tipicos` para encontrar y actualizar los testores típicos basados en las filas de la matriz.

6. Finalmente, se imprime en la consola la representación de los testores típicos encontrados en un formato específico utilizando un bucle `for` al final del código.

Esta sección del programa principal inicializa la matriz, encuentra los testores iniciales basados en la primera fila y luego utiliza la función `completa_testores_tipicos` para completar y actualizar la lista de testores típicos basándose en las filas restantes de la matriz. Finalmente, muestra los testores típicos encontrados en la consola en un formato determinado.

4. EJERCICIO 3

Matriz básica A

Para realizar la matriz básica se colocan unos y ceros de tal forma que comparando dos filas exista al menos un par de columnas que formen la matriz identidad. Esto se repite para todas las filas.

La condición gráfica de esto es que en alguna parte seamos capaces de encontrar:

1	0
0	1

Por ejemplo, si se compara la primera fila con la segunda se puede observar como X1 y X2 forman una matriz identidad con lo cual se descartan. Lo mismo sucede comparando la primera fila con la tercera fila. Sin contar la segunda fila, en X2 y X3 también existe una matriz identidad. Una vez culminada la comparación de la primera fila con todas las demás se procedió a comparar la segunda fila con el resto de filas y así sucesivamente hasta acabar con todas las comparaciones posibles.

Una vez que se hayan comparado todas las filas y en todas se han podido encontrar matrices identidad se dice que son incomparables. Es ahí cuando se ha encontrado la matriz básica de A.

X1	X2	X3	X4	X5
1	0	1	1	0
0	1	1	0	1
1	1	0	0	1
0	1	0	1	1
1	1	0	1	0

2.3.2 Conjunto de todos los testores típicos de A después de la implementación computacional del algoritmo YYC realizado en el ejercicio 2.

Aplicando exactamente el mismo método del algorítmo YCC se pudieron hallar los siguientes testores típicos de la matriz A:

- $\{x1, x2\}$
- $\{x1, x5\}$
- $\{x3, x1, x4\}$
- $\{x3, x2\}$
- $\{x4, x2\}$
- $\{x4, x5\}$

4.1. EJERCICIO 4

2.4.1 Matriz B construida a partir de matriz A

X1	X2	X3	X4	X5
1	0	1	1	0
0	1	1	0	0
1	1	0	0	0
1	0	0	0	1
0	1	0	1	0
0	0	0	1	1

La matriz B se construyó con las mismas características xj para la asignación de columnas y el número de filas se determinó a partir del número de testores típicos obtenidos en la matriz A. Para establecer los unos en cada fila y columna se tomaron en cuenta las características xj que forman cada testor típico.

Si en la primera matriz se obtuvo como testor típico $\{x3, x1, x4\}$, entonces para la primera fila de la nueva matriz B se colocaría uno (1) en las columnas X1, X3, X4 y en el resto de columnas X2, X5 se colocaría un cero (0). Lo mismo sucede con la segunda fila. Como $\{x3, x2\}$ son testores típicos de la matriz A entonces se colocarán en esas misma columnas su respectivo uno (1) y en el resto de columnas cero (0).

2.4.2 Conjunto de todos los testores típicos de B después de la implementación computacional del algoritmo YYC realizado en el ejercicio 2.

Aplicando exactamente el mismo método del algorítmo YCC se pudieron hallar los siguientes testores típicos:

- $\{x1,\ x2,\ x4\}$
- $\{x1,\ x2,\ x5\}$
- $\{x3,\ x1,\ x4\}$

- $\{x3,\ x2,\ x5\}$
- $\{x4,\ x2,\ x5\}$

2.4.3 Se puede observar que los testores típicos de la matriz B, en notación estándar, forman la matriz A. Es decir, si se intenta establecer las características xj de los nuevos testores típicos de B como las nuevas etiquetas de las columnas y rellenar las filas con uno (1) de acuerdo a sus características xj de primer testor y cero (0) en las columnas que restan, no sería posible, ya que se ha demostrado que se construye nuevamente la misma matriz A.

Dicho de otro modo, si se continúa con el proceso de la sección 2.4.1 hasta llegar a completar la nueva matriz, se llegaría a encontrar con la matriz A establecida en un inicio.

Por otro lado, se pueden determinar los testores típicos de la matriz B a partir de A con tan solo observar en que posición de las filas se encuentran unos (1). Como se conoce, a partir de los testores típicos de A se puede construir una matriz que en este caso es B. Cuando volvemos a determinar los testores típicos de B y se quiere construir una nueva matriz C, se toparía nuevamente con la matriz A. Es decir, se puede saber los testores típios de la matriz B sin tener que correr el código para averiguarlo.

5. EJERCICIO 5

Tabla 5

N	MATRIZ	FILAS	COLUMNAS	$\Psi*$	YYC	YYC ordenado
1	$\phi(\Theta(A,B))$	30	10	11	0.1954	0.2085
2	$\phi^2(\Theta(A,B))$	30	20	68	0.2525	0.2835
3	$\phi^3(\Theta(A,B))$	30	30	207	0.5539	0.6302
4	$\phi^4(\Theta(A,B))$	30	40	464	1.3790	1.6912
5	$\phi^5(\Theta(A,B))$	30	50	875	3.6280	4.3997

Tabla 6

N	MATRIZ	FILAS	COLUMNAS	$\Psi*$	YYC	YYC ordenado
1	$\gamma(\Theta(A,B))$	30	10	11	0.1954	0.2652
2	$\gamma^2(\Theta(A,B))$	60	20	121	0.3230	0.4053
3	$\gamma^3(\Theta(A,B))$	90	30	1331	3.3937	5.8228
4	$\gamma^4(\Theta(A,B))$	120	40	14641	52.8237	115.7295

6. Observaciones finales

Se pudo llegar a observar que el algoritmo YYC funcionó adecuadamente, nunca se encontró dificultad en la compilación del programa para el conjunto de experimentos en los que se trabajó. Asimismo, se pudo determinar que el tiempo de ejecución

varía segun la computadora que se esté utlizando para la compilación del programa. Sin embargo, al momento de ejecutar el algoritmo YYC en una matriz ordenada de manera ascendente, se llegó a observar un mayor tiempo de ejecución comparado al tiempo de ejecución de la matriz original.

Referencias

Alba-Cabrera, E., Ibarra-Fiallo, J., Godoy-Calderón, S., Cervantes-Alonso, F. (2014). *YYC: a fast performance cmincremental algorithm for finding typical testors.* En Lecture Notes in Computer Science (pp. 416-423). https://doi.org/10.1007/978-3-319-12568-$8_5$1

Alba-Cabrera, E., Godoy-Calderón, S., Ibarra-Fiallo, J. (2016). *Generating synthetic test matrices as a benchmark for the computational behavior of typical testor-finding cmalgorithms.* Pattern Recognition Letters, 80, 46-51. cmhttps://doi.org/10.1016/j.patrec.2016.04.020