UNIVERSIDAD AUTÓNOMA DE TLAXCALA

Facultad de Ciencias Básicas Ingeniería y Tecnología

INGENIERÍA EN COMPUTACIÓN

ACTIVIDAD:

Generación de arboles de patrones de frecuencia (FP)

PRESENTA:

DOCENTE:
M.C. Jorge Arturo Flores López

ALUMNO:
Christopher Rojano Jimenez

SEMESTRE Y GRUPO:
8 “B”

Correo:
20191414@uatx.mx

Apizaco, Tlaxcala, enero 2023

Introducción

Arboles de patrones frecuentes (FP)

Un árbol de patrones de frecuencia (FP) es un tipo de estructura de datos utilizada en minería de datos para almacenar y agrupar patrones frecuentes en grandes conjuntos de transacciones o datos de eventos. Los patrones de frecuencia son secuencias de ítems o eventos que ocurren con una cierta frecuencia en los datos.

Los principales usos de los árboles de patrones de frecuencia son:

1. Compresión de datos: permite agrupar patrones frecuentes y representarlos con un solo nodo en el árbol, reduciendo así la cantidad de información almacenada y la complejidad de los datos.
2. Descubrimiento de reglas de asociación: los patrones frecuentes
3. obtenidos a partir del árbol pueden ser utilizados para descubrir reglas de asociación entre ítems o eventos, lo que puede ser útil en aplicaciones como la recomendación de productos o la detección de fraudes.
4. Clasificación: los patrones frecuentes pueden ser utilizados como atributos para la clasificación de nuevas transacciones o eventos.
5. Predicción: los patrones frecuentes pueden ser utilizados para predecir futuros eventos o tendencias en los datos.

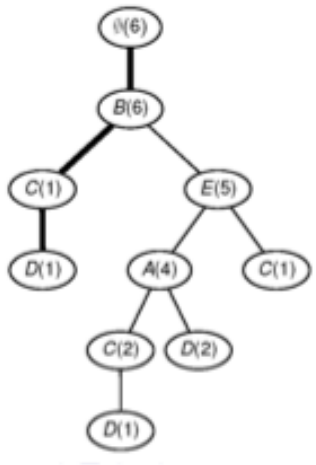

Figura 1. Árbol FP

Objetivo y resultados esperados.

Lo que el presente proyecto pretende es crear un programa en MATLAB que genere el árbol FP de un conjunto de patrones.

Para esto recibirá de forma visual las forma que se desean analizar (como números o letras), después generará ambas tablas, la tabla de transacciones y la tabla de transacciones ordenadas. Con lo anterior, gráficara el esqueleto tal cual del árbol de patrones frecuentes.

Código

Generación de cada forma en una matriz, indicando los valores.

Inicialmente se tiene la interfaz donde se ingresarán los datos, esta cuenta con una serie de botones para las diversas acciones del proyecto, además de incluir dos botones para ejecutar dos ejemplos precargados. También se tiene una matriz de botones etiquetados con letras de a-x los cuales sirven para que el usuario ingrese las formas a analizar. Finalmente esta un campo numérico para indicar el umbral de repeticiones mínimas a considerar, esto se puede observar en la figura 1.

Figura 2. Interfaz principal

Generación de la tabla de transacciones de cada letra/digito.

La matriz de botones esta compuesta de botones de estado, esto permite ingresar la forma deseada (sean números o letras) tal cual se observa en la figura 3.

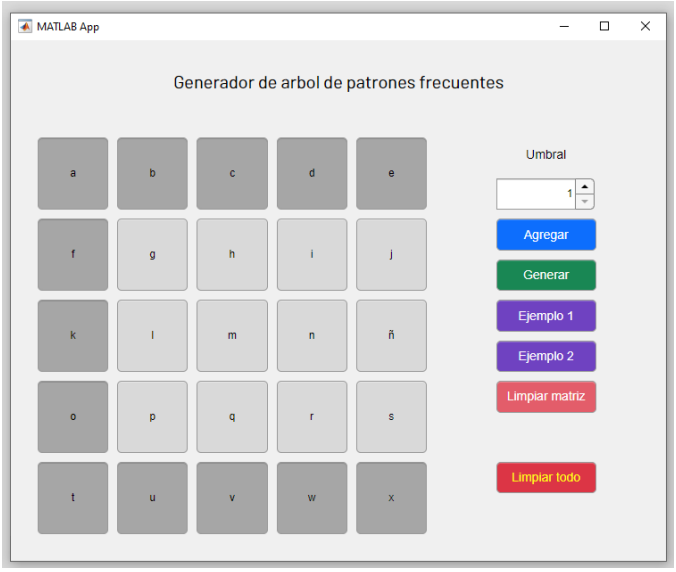

Figura 3. Ingreso de una forma

Lo siguiente es presionar el botón de “Agregar”, esto añadirá la forma a nuestra matriz de formas.

```
matAux = strings(25, 1);

if app.aButton.Value
    matAux(1) = 'a';
else
    matAux(1) = '*';
end

if app.bButton.Value
    matAux(2) = 'b';
else
    matAux(2) = '*';
end
```

Figura 4.1. Ingreso de una forma

```
matAux = matAux.';
if numel(app.matL) == 0
    app.matL = [ matAux ];
else
    app.matL = [ app.matL ; matAux];
end

app.lblConsole.Text = "Agregado correctamente!";
app.clearBtns();
```

Figura 4.2. Ingreso de una forma

Figura 4. Proceso de generación de tabla de transacciones

Dado como funciona MatLab, se lee cada botón para comprobar que el botón de cierta letra este activo y guarda dicha letra en una matriz de apoyo, de lo contrario guarda un “*” (Figura 4.1). Finalmente se agrega a la matriz global y se limpia el estado de los botones (Figura 4.2).

Generación de la tabla de transacciones ordenadas por frecuencia.

Una vez agregadas todas las formas deseadas sigue generar la tabla de frecuencias, para esto el usuario debe presionar en “Generar”, esto iniciará el proceso de generación.

```
incidencias = zeros(1,25);
for i=1: nWords
    for j=1: 25
        if find(transTbl(i:i,1:25) == mat(j))
            incidencias(j) = incidencias(j)+1;
        end
    end
end
```

Figura 5.1. Ingreso de una forma

```
newTransTbls = [ ];
for i=1: nWords
    strAx = '';
    for j=1: 25
        if mat(j)==transTbl(i,j) && incidencias(j)>umbral
            strAx = addOrderElement(mat, incidencias, strAx, j);
        end
    end
    newTransTbls = [newTransTbls strcat(strAx, "")];
end
```

Figura 5.2. Ingreso de una forma

Figura 5. Proceso de generación de tabla de transacciones

El algoritmo consiste en recorrer la tabla de transacciones “`transTbl`” y encaso de encontrar una incidencia de la letra `j` en la tabla `mat`, esta última contiene una lista ordenada de cada elemento posible (a-x), en caso de encontrarlo aumenta en uno el numero incidencias. Esto se hace como se muestra en la figura 5.1 para cada forma ingresada en el punto anterior.

Una vez con la tabla de incidencias generadas, lo que sigue es filtrar las letras que no cumpla el umbral. Un paso extra es ir agregando de forma ordenada a la nueva matriz de transacciones.

Esto ultimo se hace en la función `addOrderElement()`, la cual es de autoría propia.

```
function addOrderElement = func2(mat, incidencias, strAx, j)
    if numel(strAx) == 0
        strAx = mat(j);
    elseif numel(strAx) == 1
        if incidencias(find(mat==mat(j))) >= incidencias(find(mat==strAx)) && j < find(mat==strAx)
            strAx = strcat(mat(j),strAx);
        else
            strAx = strcat(strAx,mat(j));
        end
    else
        if incidencias(j) >= incidencias(find(mat==strAx(1))) && j < find(mat==strAx(1))
            strAx = strcat(mat(j),strAx);
        elseif incidencias(j) <= incidencias(find(mat==strAx(numel(strAx)))) && j > find(mat==strAx(numel(strAx)))
            strAx = strcat(strAx,mat(j));
        else
            inserted=0;
            for k=2: numel(strAx)
                if incidencias(j) == incidencias(find(mat==strAx(k))) && j < find(mat==strAx(k))
                    strAx = strcat(strAx(1:k), mat(j), strAx(k+1:numel(strAx)));
                    inserted=1;
                    break
                else
                    if incidencias(j) > incidencias(find(mat==strAx(k)))
                        strAx = strcat(strAx(1:k-1), mat(j), strAx(k:numel(strAx)));
                        inserted=1;
                        break
                    end
                end
            end
            if inserted==0
                strAx = strcat(strAx,mat(j));
            end
        end
    end
```

Figura 6. Función para agregar un elemento ordenado alfabéticamente y acorde al número de incidencias

El algoritmo de la Figura 6 es básicamente uno de añadir elementos a una lista ordenada, se siguen los pasos típicos de este algoritmo, adicionando dos comprobaciones, alfabéticamente y acorde al número de incidencias:

- Si la lista esta vacía, agrega el elemento,
- Si la lista tiene un elemento, comprueba si el elemento es menor y ponlo al final, de lo contrario agrégalo al inicio
- Si la lista tiene más de un elemento, si el elemento es mayor al elemento del inicio o el del final insertarlo donde corresponda, de lo contrario, recorre hasta encontrar la posición donde el elemento sea mayor, si se llego al final de la lista agregarlo al final.

Generación del esqueleto del árbol.

Finalmente para generar el árbol toca hacer una serie de pasos, el primero consiste en generar los nodos del árbol y las coordenadas de cada uno dentro de nuestra tabla de transacciones ordenada, después generar los pares de los nodos que estarán conectados, es importante también tener en cuenta los sub-nodos que se desprenden, esto tendrán un identificador distinto.

```
for i=1:nWords
    B = convertStringsToChars( newTransTbls(i) );
    y=0;

    for j=1:numel(B)-1
        nP = strcat(B(j),",",B(j+1));
        nC = strcat(num2str(x),",",num2str(y));
        nCnext = strcat(num2str(x),",",num2str(y));

        if numel(justPoint) == 0
            justPoint = [ justPoint strcat("",nP) ];
            points = [ points strcat("",nP)  ];

            nodes = [ strcat("",B(j)) ];
            nodes = [ nodes strcat("",B(j+1)) ];
            nodesCh = [ strcat(num2str(x),",",num2str(y)) ];
            nodesCh = [ nodesCh strcat(num2str(x),",",num2str(y-1)) ];
        else
            if i==1
                justPoint = [ justPoint strcat("",nP) ];
                points = [ points strcat("",nP) ];
                nodes = [ nodes strcat("",B(j+1))];
                nodesCh = [ nodesCh strcat(num2str(x),",",num2str(y-1)) ];
```

Figura 7. Generando los nodos del primer elemento de la tabla ordenada de transacciones

Para generar los nodos, coordenadas y uniones, se harán al mismo tiempo, donde nP será el punto a guardar en el arreglo `points`, el elemento nC serán las coordenadas del punto antes mencionado y se almacenará en el arreglo nodesCh, finalmente el nodo será la letra de dicho punto, inicialmente si `i` es igual a 1, como se muestra en la figura 7, se tomarán directamente los puntos, esto ya que es el primer elemento (forma) de la tabla ordenada de transacciones.

```
            else
                if numel( find( justPoint==nP ) ) == 0
                    newNodes = newNodes+1;
                    axCh = '';
                    for a=1:newNodes
                        axCh = strcat(axCh, "'");
                    end

                    points = [ points strcat(B(j),",",B(j),axCh) ];
                    nodes = [ nodes strcat("",B(j),axCh)];
                    nodesCh = [ nodesCh strcat(num2str(x),",",num2str(y)) ];

                    for k=j: numel(B)-1
                       nP = strcat(B(k),",",B(k+1));
                       nC = strcat(num2str(x),",",num2str(y));

                       justPoint = [ justPoint strcat("",nP) ];
                       points = [ points strcat(B(k),axCh,",",B(k+1),axCh) ];
                       nodes = [ nodes strcat("",B(k+1),axCh)];
                       nodesCh = [ nodesCh strcat(num2str(x-1),",",num2str(y)) ];
                       x = x-1;
                    end
                    j=k;
                end
            end
        end
        y = y-1;
    end
end
```

Figura 8. Generando los nodos para los demás elementos de la tabla ordenada de transacciones

Para los demás elementos de la tabla se sigue el proceso de la Figura 8. Se comprobará que nodo coincida con la letra que corresponde a la primera serie, por ejemplo:

a	e	t	b	c	Primera serie
a	e	t	b	c	Coincide
a	e	t	d	f	No coincide

Si no sigue la secuencia significa que se trata de una subrama del árbol, cuando es esto se cambia de agregar la letra a añadirle un “ ’ ” según el número de subrama que sea, junto con la propia letra, para la primer subrama sería a’, para la segunda a’’, etc.

En cada iteración se va aumentando el valor de x o el de y para ir generando dinámicamente las coordenadas de los distintos nodos.

Lo siguiente es generar el vector para graficar los nodos, así como la matriz de incidencias que permite agregar las uniones entre cada nodo, el proceso se muestra en la Figura 9:

```
G = zeros(numel(nodes), numel(nodes));
V = zeros(2, numel(nodes));
for i=1: numel(nodes)
    for j=1: numel(nodes)
        vertic = strcat(nodes(i),",",nodes(j));
        if find( points==vertic ) ~= 0
            G(i,j) = 1;
        end
    end
    [dot ,y] = split(nodesCh(i), ",");
    V(1,i) = str2double(dot(1));
    V(2,i) = str2double(dot(2));
end
```

Figura 9.1. Generación de nodos y matriz de incidencias

```
V = V.';
figure(100);
plot( V(:,1), V(:,2), 'ok','MarkerSize',10 );
hold on
```

Figura 9.2. Graficado de los vectores

Figura 9. Proceso de generación y graficado de vectores

Se recorren los nodos y se sacan las combinaciones, si la combinación se encuentra en la lista de puntos se agrega a la matriz en forma de una 1. Igual para cada nodo se sacan sus coordenadas y se agregan al vector de nodos. Finalmente en la figura 9.2 se grafican todos los nodos.

```
for i=1: size(G,1)
    for j=1: size(G,1)
        if G(i,j)==1
            [dot01 ,~] = split( nodesCh(find( nodes==nodes(i) )), ",");
            [dot02 ,~] = split( nodesCh(find( nodes==nodes(j) )), ",");
            arisA = [str2double(dot01(1)) str2double(dot01(2))];
            arisB = [str2double(dot02(1))  str2double(dot02(2))];

            Y = [ arisA(1:2); arisB(1:2) ];

            plot( Y(:,1), Y(:,2) ,'-r' );
            hold on
        end
    end
end

plot( [1,1], [1,1], 'ok','MarkerSize',15 );
plot( [1,1], [0,1] ,'-r' );
hold on
```

Figura 10. Graficando los vértices del árbol

En la figura 10 se detalla el proceso para generar los vértices, primero se recorre la matriz de incidencias, cuando se encuentra que hay un punto, se buscan las coordenadas de ambos nodos y se grafican.

Finalmente se agrega el nodo raíz y su conexión con el primer nodo al árbol. El resultado es el mostrado en la figura 11:

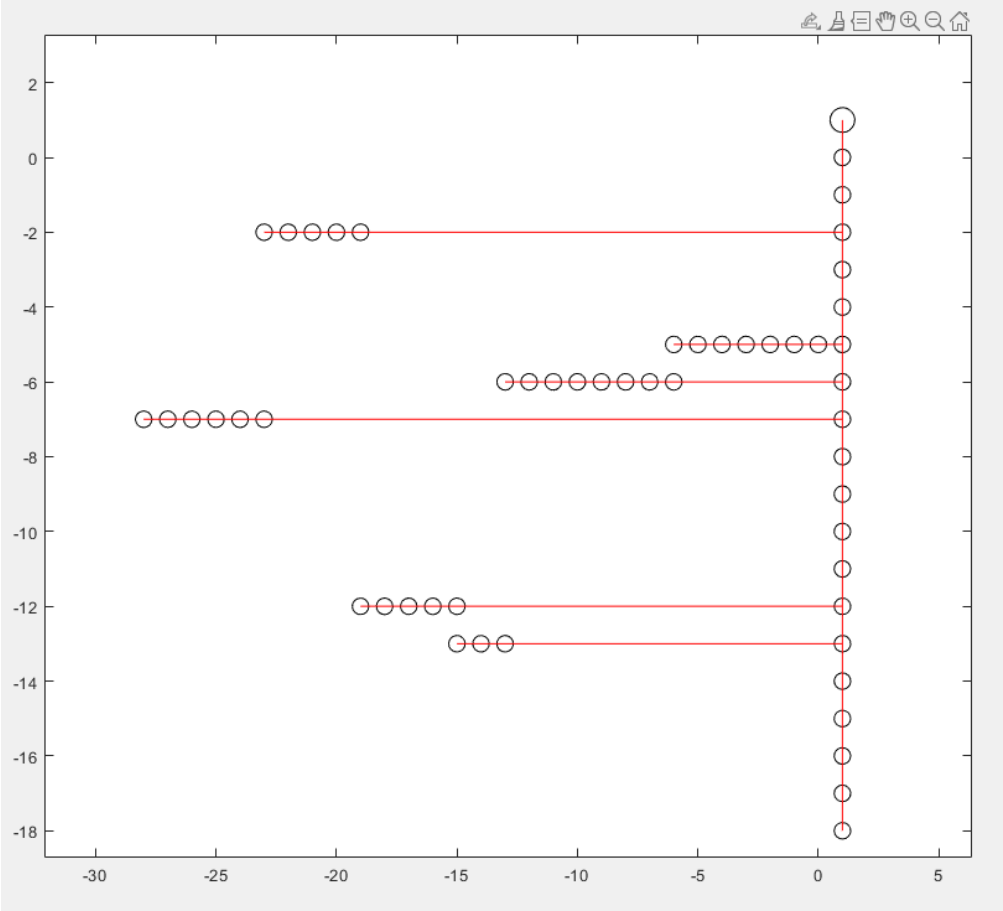

Figura 10. Graficando los vértices del árbol