Representación del conocimiento
Práctica 2

Víctor Castañeda Balmori, Mario Cuesta Rivavelarde,
Carlos García Arenal, Laro Ayesa Sánchez y Alexandru Solovei Popa

17/10/2025

La práctica consiste en programar el algoritmo EV (eliminación de variables) para una red bayesiana, tanto para inferencia marginal como condicional. Para ello, hay que estructurar la distribución de probabilidad de una red bayesiana en forma de factores e implementar el producto y marginalización de los factores.

1. Explicación de la implementación propuesta

La estructura de los factores y la forma en que transferimos la información de la red bayesiana a estos algoritmos pueden ser variables; por tanto, vamos a explicar nuestro diseño. El código se encuentra en el *practica2.zip*.

1.1. Clases Factor y Variable

Para los algoritmos hemos supuesto una lista de factores iniciales ya conocida como información de la red bayesiana.

Para gestionar los factores, primero hemos creado una clase variable, que tiene una cardinalidad (k) y unos valores. Estos valores son tuplas con el nombre de la variable y su valor de cardinalidad, es decir, la variable A con cardinalidad 2 tiene dos valores (A, 0) y (A, 1).

Una vez definidas las variables pasamos a la estructura del factor. La clase factor consiste en unas variables, y unas asignaciones. Las asignaciones serían las tablas con los valores de probabilidades del factor. De esta forma, la llave (cada fila de la tabla) es un conjunto de valores de variables. El valor de cada fila (llave) del mapa corresponde con la probabilidad de esos valores de variables en la distribución.

1.2. Marginalización

Para marginalizar un factor respecto a una variable hay que seguir el algoritmo visto en clase. Creamos un factor τ y recorremos toda la tabla (el mapa de asignaciones) del factor. Para cada asignación, eliminamos la variable que queremos marginalizar y añadimos al factor τ el valor. Si el factor τ ya tiene una asignación con esa llave, lo sumamos. De esta forma, creamos el τ como factor marginalizado.

1.3. Producto de factores

Aquí aplicamos una idea similar a la de la marginalización y seguimos el procedimiento visto en clase. Creamos un factor vacío con las asignaciones que tendrá el factor resultante del producto. Esto es, teniendo en cuenta si tienen variables en común o no, crear las combinaciones de las variables con sus respectivas cardinalidades. Después, recorremos las asignaciones del factor resultante y buscamos en las llaves de los factores que se multiplican, las asignaciones que debemos operar. El hecho de tener que recorrer las variables de las asignaciones introduce una mayor complejidad a la vista en clase.

1.4. Inferencias

Una vez diseñada la estructura de los factores, el producto y la marginalización, la implementación de EV para las inferencias es trivial. Pero, cabe destacar algún detalle.

- **Inferencia marginal:** En lugar de pasar las variables que vamos a mantener, pasamos el orden de eliminación del resto. Es decir, si queremos p(a,b) a partir de p(a,b,c,d,e), a la función inferencia-marginal le pasamos $orden = \{d, c, e\}$ por ejemplo. Asumimos que el orden óptimo ya lo hemos calculado. Además, pasamos todos los factores de la red bayesiana.

- **Inferencia condicional:** De nuevo, pasamos el orden de eliminación de variables en el numerador y en el denominador (teorema de Bayes).
  Dado el orden de eliminación de variables del numerador, usamos la inferencia marginal para obtener el factor del numerador. Después, usamos ese factor calculado para, con el orden del denominador, calcular el factor del denominador. Una vez hecho esto, multiplicamos el factor del numerador por el inverso del factor del denominador y obtenemos una tabla con las probabilidades condicionales buscadas. No hemos hecho un algoritmo para valores concretos, usamos el general y filtramos. De esta forma, para hacer $p(a \mid b) = p(a, b)/p(b)$ en p(a,b,c,d,e), pasaríamos orden-numerador = {c,d,e}, orden-denomimador = {a} y los factores de la red bayesiana.

2. Evaluación del coste temporal

2.1. Teórico

Tal y como hemos visto en clase, la complejidad asintótica temporal del algoritmo de eliminación de variables es $O(n * k^{l+1})$ siendo n el número de variables o nodos de la red bayesiana, k la cardinalidad de las variables y l el número de variables del τ con mayor número de variables. Una vez programado el algoritmo, vemos que es coherente con lo estudiado, pero que nuestra implementación es un poco menos eficiente.

La marginalización de factores recorre todas las asignaciones de un factor, por lo que su complejidad es $O(k^{n_f})$ siendo n_f el número de variables del vector y k la cardinalidad de las variables.

El producto de factores crea las asignaciones del factor resultante y las recorre, rellenando las filas. Dado que debemos recorrer las variables de cada asignación para rellenar las filas, la complejidad de nuestro algoritmo es peor que la teórica. La complejidad es, $O(n_1 \bigcup n_2 * k^{n_1 \bigcup n_2})$. También tenemos una versión cuya complejidad era $O(k^{n_1+n_2})$, pues recorría todas las asignaciones de los 2 factores a multiplicar (no la entregamos, es peor).

El algoritmo de EV para inferencia marginal recorre las variables en el orden de eliminación, realiza multiplicaciones de factores y la marginalización de los factores resultantes. En cada iteración se reduce en un factor el problema, y cada factor es un nodo (variable) de la red bayesiana. O(n) siendo n el número de variables de la red.
Además, el producto de variables y la marginalización se hace en cada iteración. Cabe mencionar que, en la implementación eficiente la complejidad asintótica de ambos es la misma, porque la marginalización se hace sobre el factor resultante del producto de factores. Si el factor resultante tiene un ámbito de m variables, la complejidad del producto y de la marginalización será k^m. $O(k^m + k^m) = O(k^m)$. Por lo tanto, la complejidad del algoritmo es $O(n * k^{max(m)})$ porque dependerá del ámbito que tenga el factor que se marginalice con mayor número de variables.

Sin embargo, en nuestra implementación el producto de factores tiene peor complejidad que la marginalización, $O(m * k^m)$ frente a $O(k^m)$. Siendo m el número de variables del τ con mayor número de variables + 1. De forma, que la complejidad de nuestro algoritmo será algo así $O(n * m * k^m)$ siendo n el número de variables, k la cardinalidad y m lo que ya he mencionado.

2.2. Empírico

En el fichero *graficas.py* se encuentra el código necesario para obtener las gráficas que se exponen a continuación. Para ejecutar cada caso estudiado es necesario modificar las variables *pruebas* y *cardinalidad*.

2.2.1. Complejidad del Algoritmo

Para comprobar la complejidad hemos usado una red bayesiana en la que X_i es padre de X_{i+1} a X_n, es decir, cada nodo es padre de todos sus siguientes.

El ejemplo calcula $p(X_n|X_0)$ con un orden de eliminación de X_1 a X_{n-1}. En este caso, primero se hará el producto de todos los factores menos el que contiene solo a X_0, y tras ello se marginalizará cada variable. Cada variable tiene la misma cardinalidad, en este caso 2.

La forma del grafo hace que haya factores que dependen de n variables (ϕ_n) y, por tanto, la eliminación de variables sea muy ineficiente. La siguiente gráfica muestra los resultados:

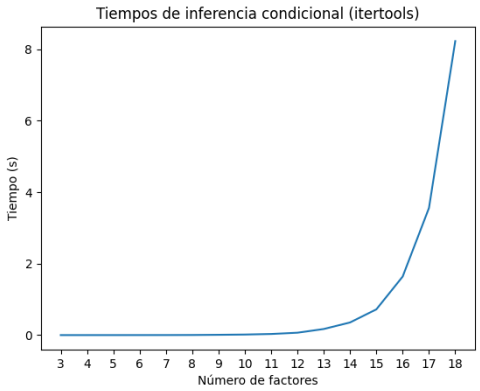

Figura 1: Complejidad Algortimo

Se puede observar que la complejidad es exponencial, siendo la complejidad de la marginalización k^n. Esta gráfica es la obtenida al ejecutar el fichero *graficas.py* con $pruebas = 0$ y $cardinalidad = 2$

2.2.2. Cambios de cardinalidad

Ahora vamos a comprobar como afecta la cardinalidad en el ejemplo anterior:

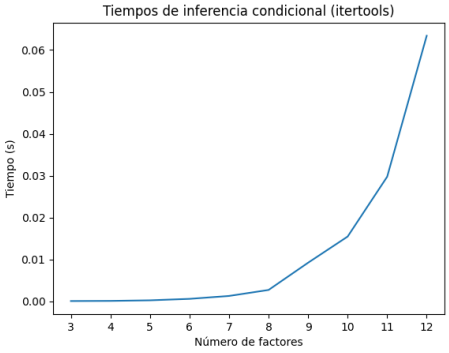

Figura 2: Cardinalidad 2

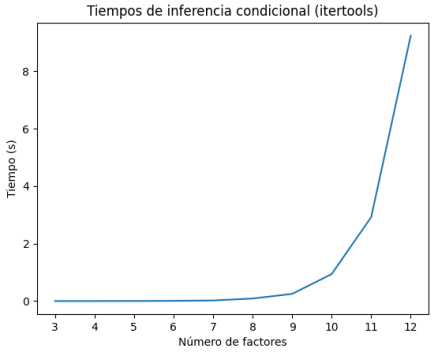

Figura 3: Cardinalidad 2

Figura 4: Comparacion cardinalidades

Las gráficas concuerdan con la complejidad del algoritmo, por ejemplo, la relación entre $3^{12}/2^{12} = 129{,}746$. El tiempo para 12 variables con cardinalidad 2 está entorno a 0,065 segundos y $0{,}065 * 129{,}746 = 8{,}43$ segundos. Esto se aproxima al tiempo que se tarda con cardinalidad 3.

2.2.3. Complejidad en función del orden de eliminación

Por ultimo, vamos a comprobar como el orden de eliminación influye en la complejidad del algoritmo. Para este ejemplo, hemos usado una red con los filas, X e Y. Los nodos de X son padres de todos los nodos de Y, mientras que los nodos de Y son padres de su siguiente, es decir, Y_i es padre unicamente de Y_{i+1}.

Para el ejemplo, hemos usado X con 2 nodos fijos, mientras que Y es el que varía en número de nodos. De esta manera, usando la heurística de mínimos vecinos, el mejor orden será primero los nodos de Y, que tendrán como vecinos los nodos de X, su anterior y posterior, lo que hace un máximo de 4 vecinos. Mientras que los nodos de X tienen como vecinos todos los nodos de Y.

Para el ejemplo se calcula $p(Y_n|Y_0)$, el resultado es el siguiente:

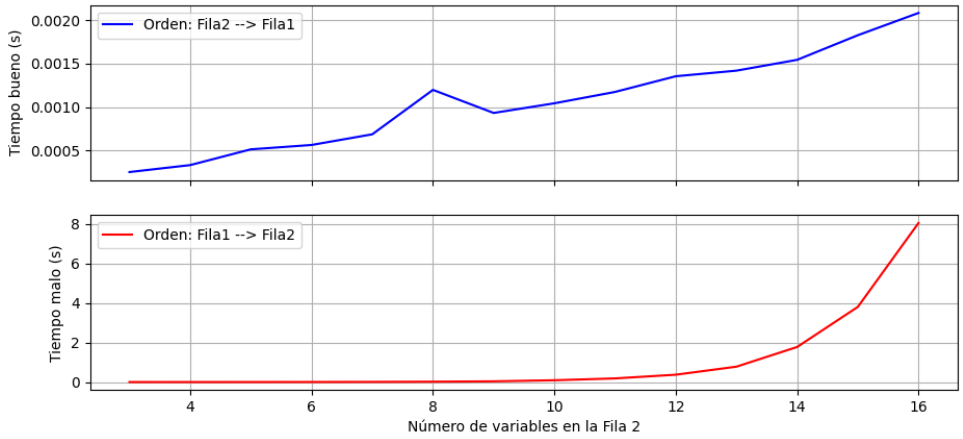

Figura 5: Comparacion Orden de Eliminacion

Se puede observar que con el primer orden la complejidad es lineal, los factores no superan las cuatro variables, la primera eliminación es sobre Y_1 que estará en dos factores $\phi(X_0, X_1, Y_0, Y_1)$ y $\phi(X_0, X_1, Y_1, Y_2)$. El producto resultará en $\varphi(X_0, X_1, Y_0, Y_1, Y_2)$ y al marginalizar queda $\tau(X_0, X_1, Y_0, Y_2)$.

Ahora se hará sobre Y_2, que estará en $\tau(X_0, X_1, Y_0, Y_2)$ y $\phi(X_0, X_1, Y_2, Y_3)$. De esta manera, llegaremos a $\tau(X_0, X_1, Y_0, Y_n)$ tras eliminar Y_{n-1}. Finalmente, eliminaremos X_0 y X_1.

Si eliminamos la fila X_0 en primer lugar, nos quedará un producto de todos los factores excepto $\phi(X_1)$, resultando en $\tau(Y_0, Y_1, ..., Y_n)$. Teniendo n variables en el factor resultante. Para obtener esta comparación hay que ejecutar el fichero con $pruebas = 1$.

3. Eficacia de la respuesta

Para comprobar la eficacia de nuestra implementación hemos decidido ejecutar el algoritmo para calcular lo que se pide en el ejercicio 1 de la hoja de problemas 1 con la red bayesiana de la Figure 6:

En esta red bayesiana se pide calcular $p(e^0 \mid c^1, d^1, f^1)$ y $p(e^1 \mid c^1, d^1, f^1)$. El cálculo teórico de estos datos nos daba:

$$p(e^0 \mid c^1, d^1, f^1) = 0{,}54$$

$$p(e^1 \mid c^1, d^1, f^1) = 0{,}46$$

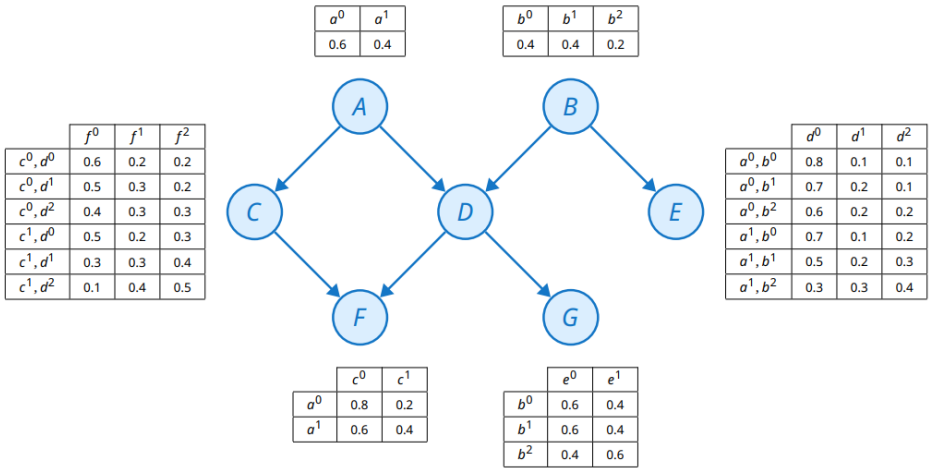

Figura 6: Red bayesiana del ejercicio

Ejecutando el código implementado obtenemos el mismo resultado, por lo que suponemos que funciona bien. El fichero donde se encuentra la creación de los factores y la ejecución del algoritmo con esa red bayesiana (Figure 6) es *practica2.py*. Es el mismo fichero donde se encuentran las clases $Variable$ y $Factor$.