Universidad Politécnica
de Madrid

Escuela Técnica Superior de Ingenieros Informáticos

Grado en Ingeniería Informática

Trabajo Fin de Grado

Análisis e Implementación del Operador de Bayes Multivariante en Redes Bayesianas, Marginalización Aproximada Vía Simulación

Autor: Ignacio Layo González
Tutor(a): Juan Antonio Fernández del Pozo de Salamanca

Madrid, junio 2022

Este Trabajo Fin de Grado se ha depositado en la ETSI Informáticos de la Universidad Politécnica de Madrid para su defensa.

Trabajo Fin de Grado
Grado en Ingeniería Informática

Título: Análisis e Implementación del Operador de Bayes Multivariante en Redes Bayesianas, Marginalización Aproximada Vía Simulación

junio 2022

Autor: Ignacio Layo González
Tutor: Juan Antonio Fernández del Pozo de Salamanca
Departamento de Inteligencia Artificial
ETSI Informáticos
Universidad Politécnica de Madrid

Resumen

Las redes Bayesianas son un modelo gráfico probabilístico que permite representar relaciones de dependencia condicional en datos multivariantes. Este modelo factoriza una distribución de probabilidad conjunta como un producto de distribuciones de probabilidad condicionada, haciendo uso de la independencia condicional entre variables en la red. Obtener la distribución de probabilidad conjunta de la red a partir de las distribuciones de probabilidad condicionadas transforma una distribución de alta dimensión en varias distribuciones marginales de baja dimensión. Este proceso de marginalización se realiza mediante inferencia exacta o aproximada.

En la actualidad se utilizan modelos probabilísticos muy complejos que hacen de la inferencia exacta un problema computacionalmente intratable. La inferencia aproximada permite abordar el problema de la inferencia en redes Bayesianas renunciando a cierta precisión. Este trabajo desarrolla los conceptos necesarios para entender las redes Bayesianas e inferencia en redes Bayesianas, así como algunos algoritmos que realizan inferencia exacta y aproximada. Finalmente se hace uso de las librerías bnlearn y gRain en el entorno de programación R para evaluar el rendimiento de dos algoritmos de inferencia aproximada en términos de tiempo de convergencia y error comparando los resultados con los valores obtenidos mediante inferencia exacta.

Abstract

Bayesian networks are a graphic probabilistic model which allows to represent condicional dependence relations in multivariate data. This model factors a joint probability distribution as a product of joint probability distributions, making use of conditional independence between variables in the net. Obtaining the joint probability distribution of the network from the conditional probability distributions transforms a high-dimensional probability distribution into several low-dimensional marginal distributions. This marginalization proccess is done using exact or approximate inference.

Currently very complex probabilistic models are used that make exact inference a computationally intractable problem. Approximate inference allows us to approach the Bayesian networks inference problem giving up some precision. This work develops the necessary concepts to understand Bayesian networks and inference in Bayesian networks, as well as some algorithms that perform exact and approximate inference. Finally, two approximate inference algorithms are evaluated using the bnlearn and gRain libraries within the R programming environment, considering their performance in terms of time of convergence and error, comparing the results with the values obtained through exact inference.

Tabla de contenidos

Capítulo 1

Introducción

En un mundo en el que cada vez existen más datos a los que podemos acceder, surgen nuevos modelos para resolver problemas con un número elevado de variables y relaciones complejas entre ellas. Las Redes Bayesianas son una potente herramienta estadística que surge en el campo de la Inteligencia Artificial que nos permite afrontar problemas de estas características.

Judea Pearl está considerado como el primer autor que utiliza el término de *Red Bayesiana* [1], aunque este modelo de representar relaciones entre variables de forma dirigida aparece previamente en publicaciones como *The Method of Path Coefficients*, de Sewall Wright, como una manera de relacionar los coeficientes de correlación entre variables y las relaciones funcionales entre ellos [2]. En la época más reciente (2009) Pearl ha publicado un libro [3] en el que describe formalmente la teoría de la causalidad, aunque lo hace sin hacer uso de este tipo de estructuras.

Tradicionalmente estos modelos han sido construidos manualmente, ayudándose de un conocimiento experto en las variables del problema para definir la probabilidad condicionada de las variables presentes en el problema. En los últimos años se han desarrollado diversas técnicas que permiten el aprendizaje a partir de cierto volumen de datos, modificando de esta manera tanto las probabilidades condicionadas como la estructura de la red.

Las Redes Bayesianas son un modelo gráfico probabilístico que representa un conjunto de variables aleatorias y las relaciones que existen entre ellas. En estas relaciones puede existir causalidad o cierta independencia condicional. A partir de este modelo gráfico podemos obtener una serie de funciones de probabilidad condicionada, que factorizan de forma más sencilla la función de probabilidad conjunta de la red. Esta herramienta nos permite abordar problemas en los que existe incertidumbre, donde las conclusiones no se pueden construir sólo con un conocimiento previo sobre el tema. Las Redes Bayesianas proporcionan una forma conveniente y coherente de representar la incertidumbre en modelos inciertos y se utilizan cada vez más para representar el conocimiento incierto [4].

Una Red Bayesiana puede representarse gráficamente como un grafo dirigido

acíclico. Cada nodo de la red representa una variable aleatoria del problema y cada vértice representa una relación de dependencia condicional entre variables. En los siguientes capítulos trataremos estos conceptos con mayor profundidad. En este trabajo se ahondará en el concepto de Redes Bayesianas discretas como modelo gráfico, aunque existen otros modelos que veremos en el capítulo 3.

Como veremos, aún siendo un modelo matemático determinista, resolver con exactitud la distribución de probabilidad conjunta de las variables que componen la red es una tarea computacionalmente muy complicada. Sin embargo, existen algoritmos aproximados que pueden resolver esta distribución en un tiempo razonable renunciando a cierta precisión en el resultado.

1.1. Estructura

El documento está estructurado de la siguiente manera: el capítulo 2 está dedicado al desarrollo del estado del arte de Redes Bayesianas Discretas, el objetivo final de estudio de este trabajo. En el capítulo 3 se mencionan otros modelos gráficos probabilísticos exponiendo las diferencias con las Redes Bayesianas discretas. Finalmente, el capítulo 4 desarrolla la teoría sobre los algoritmos de inferencia que van a ser estudiados y evaluados, y se expondrán los resultados obtenidos por estos algoritmos.

Capítulo 2

Redes Bayesianas Discretas

En este capítulo se estudiará e investigará sobre el estado del arte de Redes Bayesianas discretas, su definición, representación gráfica y algunos conceptos relevantes; e inferencia y marginalización, como mecanismo para estimar valores de las variables de la red y sus probabilidades.

Una red Bayesiana de un conjunto de variables $X = \{X_1, ..., X_n\}$ representa una distribución de probabilidad conjunta sobre esas variables [5]. Se representa gráficamente como un grafo dirigido acíclico. Sus nodos representan las variables aleatorias y sus aristas las dependencias de probabilidad entre las variables. De este modo, los nodos inconexos representan variables independientes unas de otras.

Formalmente, una red Bayesiana es un par $B = (G, \theta)$, donde $G = (V, A)$ es un grafo dirigido acíclico con un conjunto de nodos o vértices $V = \{1, ..., n\}$ y un conjunto de aristas o arcos $A \subseteq \{(a, b) \in V \times V : a \neq b\}$. Es decir, pares de nodos (a, b) pertenecientes a V tal que a es el nodo inicial de la arista y b es el nodo final. $\theta = \{P(x_i|x_{Pa(i)}), i = 1, ..., n\}$ es un conjunto de parámetros que define la distribución de probabilidad condicional para cada variable de la red, donde $Pa(i)$ es el conjunto de nodos padre del nodo i. Una red Bayesiana [6] factoriza una distribución de probabilidad conjunta $P(x)$ de un vector de variables aleatorias $X = (X_1, ..., X_n)$. Cada variable X_i es condicionalmente independiente de los nodos no descendientes de la misma dado $X_{Pa(i)}$. Esta propiedad se conoce como propiedad local de Markov. El conjunto de nodos descendientes de i es el conjunto de todos los nodos que se pueden alcanzar siguiendo un camino directo a través de los arcos dirigidos desde i.

En el ejemplo de la figura 2.1 podemos ver un ejemplo sencillo de una red Bayesiana. Tenemos la variable "WetGrass"(W) que tiene dos causas posibles: "Sprinkler"(S) o Rain"(R). Las tablas de cada nodo representan la probabilidad condicionada para esa variable. Por ejemplo, en el nodo "WetGrass"podemos ver que:

$$Pr(W = T|S = T, R = F) = 0.9$$

Y por lo tanto:

$$Pr(W = F|S = T, R = F) = 1 - 0.9 = 0.1$$

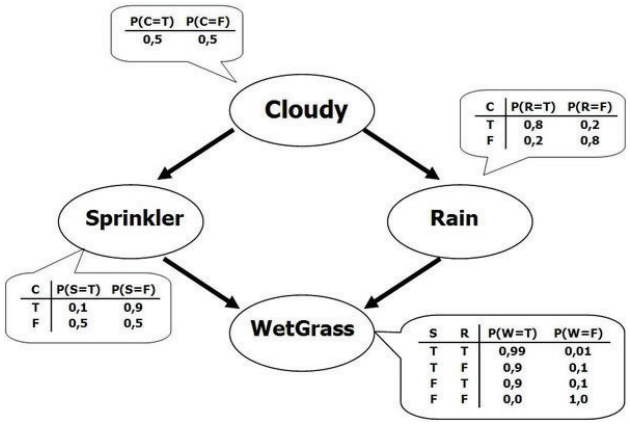


Figura 2.1: Pequeño ejemplo de una red Bayesiana [7].

Es decir, la probabilidad de que la hierba esté mojada (W=T) condicionado a que el aspersor está encendido (S=T) y no llueve (R=F) es de 0.9. Por lo tanto la probabilidad de que la hierba esté seca en las mismas condiciones es de 0.1.

El nodo Çloudy"(C) no tiene ninguna arista dirigida hacia él (no tiene nodos padre), así que su tabla no representa una probabilidad condicionada, sino la probabilidad a priori de los posibles valores para esta variable. En este caso, $P(C = T) = 0.5$ y por ende $P(C = F) = 0.5$.

Por la regla de la cadena, la probabilidad condicionada de todos los nodos es:

$$P(C, S, R, W) = P(C) * P(S|C) * P(R|C, S) * P(W|C, S, R) \tag{2.1}$$

Sin embargo, podemos simplificar el tercer término porque la variable R es independiente de S en presencia de su nodo padre C; y el último término porque W es independiente de C dados sus nodos padres S y R. La probabilidad condicionada de todos los nodos queda de la siguiente manera:

$$P(C, S, R, W) = P(C) * P(S|C) * P(R|C) * P(W|S, R) \tag{2.2}$$

De este modo comprobamos que las relaciones de independencia entre los nodos nos permiten representar la probabilidad condicionada de una forma más compacta [8].

Cada variable en una red Bayesiana es independiente de sus ancestros dadas las probabilidades condicionadas de sus padres [5]. Tomando como referencia la figura 2.1, la variable "WetGrass.$^{\text{es}}$ independiente de la variable Çloudy"dados "Sprinklerz Rain".

2.1. D-Separación

Una red Bayesiana puede utilizarse para observar cómo un cambio en la certeza de una variable puede cambiar la certeza de otra variable.

Consideramos el ejemplo de la figura 2.2. El nodo *A* es padre de *B*, que a su vez es padre de *C*. Obtener cierta evidencia sobre *A* influirá sobre la certeza de

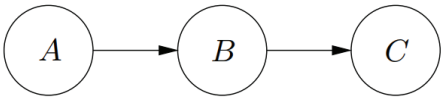


Figura 2.2: Conexión en serie. Si *B* está instanciado, corta la comunicación entre *A* y *C*.

B, que a su vez influye sobre la certeza de *C*. Esto también ocurre en el sentido inverso; la evidencia de *C* influye en la certeza de *A* a través de *B*. Sin embargo, si el estado de *B* es conocido, *A* y *C* se vuelven independientes; decimos que *A* y *C* están d-separadas dado *B*. Cuando se conoce el estado de una variable, se dice que la variable está *instanciada* [9] con cierta evidencia.

Figura 2.3: Conexión en serie para *Nublado*, *Lluvia* y *Hierba húmeda*.

La figura 2.3 muestra una red Bayesiana para las relaciones entre *Nublado (nada, poco, mucho)*, *Lluvia (nada, poco, mucho)* y *Hierba Húmeda (sí, no)*. Si no he observado si llueve o no, saber que la hierba está húmeda aumenta la evidencia de lluvia, que a su vez tiene un efecto sobre la evidencia de las nubes. El mismo razonamiento se puede aplicar en orden inverso. Sin embargo, si conozco el nivel de lluvia, saber que la hierba está húmeda no me aporta información sobre las nubes.

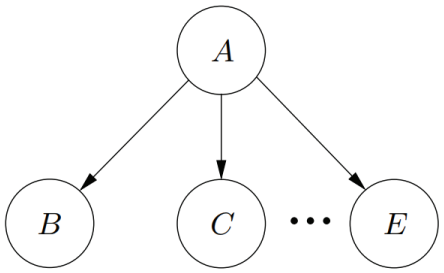


Figura 2.4: Conexión divergente. Si *A* está instanciado, corta la comunicación con sus nodos hijos.

La red mostrada en la figura 2.4 es una conexión divergente. La evidencia puede propagarse entre todos los hijos de *A* salvo que *A* esté instanciado.

En el caso de la figura 2.5, si no sabemos si la temperatura es baja o alta, saber que hay heladas nos da información sobre la temperatura, que a su vez tendrá un impacto en la evidencia de resfriado. Sin embargo, si conocemos la temperatura, saber que hay hielo no nos aporta nueva información sobre los resfriados.

La situación de la figura 2.6 es una conexión convergente. En este caso, si no se sabe nada de *A* excepto lo que se puede inferir por sus padres *B*,...,*E* entonces

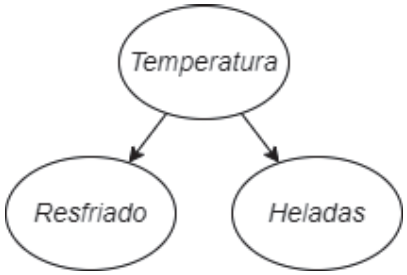


Figura 2.5: Ejemplo de conexión divergente para *Temperatura*, *Resfriado* y *Heladas*.

sus padres son independientes. Conocer una posible causa de un evento no nos da información sobre el resto de causas posibles. Veámoslo con un ejemplo.

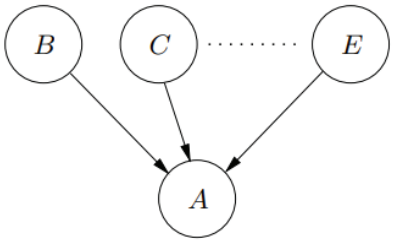


Figura 2.6: Conexión convergente. Si *A* cambia su certeza, se abre la comunicación entre sus padres.

Tomamos como referencia el caso de la figura 2.7. Si no tenemos ninguna evidencia sobre el estado de *hierba húmeda*, saber que el aspersor está encendido no nos da ninguna información sobre la lluvia. Sin embargo, si sabemos que la hierba está húmeda y comprobamos que no llueve, aumenta la evidencia de que el aspersor esté encendido.

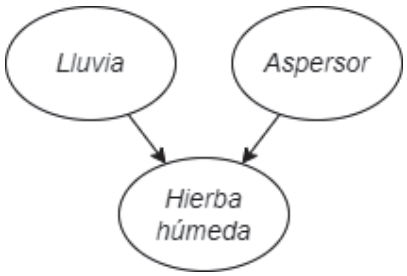


Figura 2.7: Conexión convergente para *lluvia*, *aspersor* y *hierba húmeda*.

Hemos visto con tres ejemplos todas las formas en las que se puede propagar evidencia a través de una variable (nodo). La siguiente definición nos da una regla para saber si dos variables son independientes dada cierta evidencia introducida en la red [9].

Dos variables distintas A y B en una red Bayesiana están ***d-separadas*** *(son condicionalmente independientes) si para todos los caminos entre A y B, existe*

una variable intermedia V diferente de A y B tal que

- *la conexión es en serie o divergente y V está instanciado.*

o bien

- *la conexión es convergente y ni V ni ninguno de sus descendientes han recibido evidencia.*

2.2. Markov Blanket

Un concepto importante relacionado con el concepto de independencia condicional entre variables de una red Bayesiana es el *manto de Markov* o *Markov blanket*. El manto de Markov se define para un nodo concreto del grafo.

El manto de Markov [10] de un nodo es el conjunto de nodos que provocan su d-separación respecto al resto del grafo. En cualquier red Bayesiana el manto de Markov de un nodo A es el conjunto de los padres de A, los hijos de A y los padres de los nodos hijos de A.

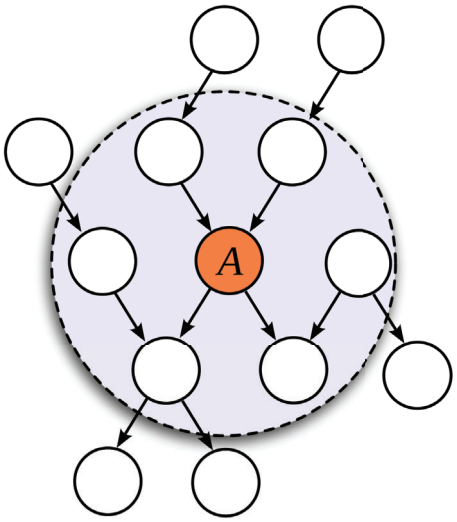


Figura 2.8: Manto de Markov del nodo A, formado por sus padres, hijos y los otros padres de sus nodos hijos.

2.3. Inferencia y marginalización

La incertidumbre es una parte inherente a los problemas de decisión, predicción, causalidad y razonamiento. La inferencia estadística estudia el razonamiento aproximado, que nos permite una capacidad de razonamiento en problemas que involucran incertidumbre, como es el caso de las Redes Bayesianas, en las que conocer una premisa no implica una certeza absoluta sobre sus implicaciones, sino ciertas probabilidades de que algunas variables sean ciertas.

La *inferencia Bayesiana* es un método de inferencia estadística que utiliza el

Teorema de Bayes para calcular estas probabilidades a partir de ciertas premisas, que pueden ser la evidencia introducida en la red y las probabilidades calculadas a partir de nodos padres de la red.

Una red Bayesiana nos da una distribución de probabilidad conjunta para todas las variables que pertenecen al problema. Dada esta distribución, se pueden resolver consultas del tipo $P(X = x)$ mediante marginalización de la distribución conjunta. Sin embargo, la tabla de probabilidad conjunta tiene un tamaño de $\prod_{i=1}^{n} p_i$, donde n es el número de nodos de la red y p_i es el número de posibles estados para el nodo i. Es decir, para una red con 2 posibles resultados por nodo la distribución de probabilidad conjunta tiene un tamaño de 2^n.

2.3.1. Inferencia exacta

Consideramos una vez más la red de la figura 2.1, y suponemos que hemos observado el estado del césped y la hierba está húmeda, es decir, $W = T$. Existen dos posibles causas: el aspersor está encendido ($S = T$) o está lloviendo ($R = T$). Vamos a calcular la probabilidad de cada uno de estos eventos. El teorema de Bayes expresa lo siguiente:

$$P(X|y) = \frac{P(y|X) * P(X)}{P(y)} \tag{2.3}$$

Donde X es un nodo para el que no se conoce su estado e y es la evidencia introducida en la red. Esta expresión es equivalente a la siguiente:

$$P(X|y) = \frac{P(X, y)}{P(B)} \tag{2.4}$$

Utilizando la ecuación (2.4) tenemos:

$$P(S = T|W = T) = \frac{P(S = T, W = T)}{P(W = T)} = \frac{\sum_{c,r} P(C = c, S = T, R = r, W = T)}{P(W = T)} = \frac{0.2781}{0.6471}$$

$$= 0.430$$

Análogamente

$$P(R = T|W = T) = \frac{P(R = T, W = T)}{P(W = T)} = \frac{\sum_{c,s} P(C = c, S = s, R = T, W = T)}{P(W = T)} = \frac{0.4581}{0.6471}$$

$$= 0.708$$

donde

$$P(W = T) = \sum_{c,s,r} P(C = c, S = s, R = r, W = T) = 0.6471$$

es una *constante de normalización* [8] para que la distribución de probabilidad $p(X|y)$ sume 1.

Utilizando el teorema de Bayes (2.3):

$$P(S = T|W = T) = \frac{P(W = T|S = T) * P(S = T)}{P(W = T)}$$

Y mediante el *Teorema de la probabilidad total* [11]:

$$\begin{aligned} P(W=T) =& P(W=T|S=T,R=T)*P(S=T,R=T) \\ &+ P(W=T|S=F,R=T)*P(S=F,R=T) \\ &+ P(W=T|S=T,R=F)*P(S=T,R=F) \\ &+ P(W=T|S=F,R=F)*P(S=F,R=F) \end{aligned}$$

La inferencia que se ha llevado a cabo para calcular estas probabilidades es exacta. En la mayoría de los casos, dejando a un lado redes muy sencillas, calcular probabilidades a posteriori mediante la regla de Bayes es un problema intratable. Para este caso en concreto, el cálculo de esta constante ($P(W=T)$) tiene que considerar cada uno de los escenarios a los que está condicionada la probabilidad de la variable en cuestión. Es decir, tiene en cuenta el escenario $Cloudy=T$, $Cloudy=F$, $Sprinkler=T$, $Sprinkler=F$, $Rain=T$ y $Rain=F$.

Se introduce a continuación la *eliminación de variables*, un método que utiliza las asunciones de independencia condicional entre variables para acelerar el cálculo vía inferencia exacta.

2.3.1.1. Eliminación de variables

Si consideramos el problema de cálculo de la anteriormente llamada *constante de normalización*, y simplificando los términos obtenidos por la regla de la cadena tenemos:

$$P(W=w) = \sum_c \sum_s \sum_r P(C=c, S=s, R=r, W=w)$$

$$= \sum_c \sum_s \sum_r P(C=c) \times P(S=s|C=c) \times P(R=r|C=c) \times P(W=w|S=s,R=r)$$

es decir, cada nodo es dependiente de sus nodos padres, no de otros nodos ancestros. La clave de este algoritmo es llevar las sumas lo más atrás posible, de la siguiente manera:

$$P(W=w) = \sum_c P(C=c) \sum_s P(S=s|C=c) \sum_r P(R=r|C=c) \times P(W=w|S=s,R=r)$$

Ahora, al realizar la suma más interna tenemos:

$$P(W=w) = \sum_c P(C=c) \sum_s P(S=s|C=c) \times \tau_1(c,w,s)$$

De este modo hemos creado un término τ_1 que no depende de la variable de la suma más interna:

$$\tau_1(c,w,s) = \sum_r P(R=r|C=c) \times P(W=w|S=s,R=r)$$

Ahora al realizar la suma más interna que queda tenemos

$$P(W=w) = \sum_c P(C=c) \times \tau_2(c,w)$$

donde

$$\tau_2(c, w) = \sum_s P(S = s|C = c) \times \tau_1(c, w, s)$$

```
Procedure Sum-Product-VE (
    Φ,    // Set of factors
    Z,    // Set of variables to be eliminated
    ≺     // Ordering on Z
)
1   Let Z_1, ..., Z_k be an ordering of Z such that
2     Z_i ≺ Z_j if and only if i < j
3   for i = 1, ..., k
4     Φ ← Sum-Product-Eliminate-Var(Φ, Z_i)
5   φ* ← ∏_{φ∈Φ} φ
6   return φ*

Procedure Sum-Product-Eliminate-Var (
    Φ,    // Set of factors
    Z     // Variable to be eliminated
)
1   Φ' ← {φ ∈ Φ : Z ∈ Scope[φ]}
2   Φ'' ← Φ − Φ'
3   ψ ← ∏_{φ∈Φ'} φ
4   τ ← ∑_Z ψ
5   return Φ'' ∪ {τ}
```

Figura 2.9: Generalización del algoritmo de eliminación de variables [12].

En general, en una cadena de nodos de longitud n, estos términos se calcularían un número exponencial de veces. Almacenando estos términos τ_i evitamos tener que calcular repetidamente estos términos. De esta manera, reducimos la complejidad de la inferencia exacta basándonos en dos ideas [12]:

1. Debido a la estructura de una red Bayesiana, algunas subexpresiones en la red sólo dependen de un pequeño número de variables. Calculando estos términos .[el]iminamos.[es]tas variables del cálculo.

2. Calculando estas expresiones una vez y almacenando los resultados, evitamos generar estos términos un número exponencial de veces.

2.3.2. Inferencia aproximada

A pesar de las ventajas de la eliminación de variables para resolver el problema de la inferencia, la inferencia exacta sigue siendo un problema que escala muy mal con el tamaño de la red. De hecho, el problema de inferencia en redes Bayesianas es *NP-hard* [13] o NP-complejo. Surge la idea de la inferencia aproximada como un modo de reducir drásticamente el tiempo de cálculo del problema de la inferencia a cambio de reducir la precisión de la misma.

Existen multitud de algoritmos que implementan inferencia aproximada. Veamos en qué consisten.

2.3.2.1. Muestreo directo

La solución más sencilla a este problema es el *muestreo directo* [14]. Los algoritmos de muestreo generan una serie de muestras aleatorias a partir de la distribución de probabilidad de la red. Estas muestras se generan siguiendo un orden topológico de los nodos de la red. Es decir, para cada muestra primero se tienen en cuenta los nodos que no tienen nodos padre, a continuación los hijos de estos nodos y así sucesivamente. Cada una de estas muestras es un caso posible para los valores de las variables de la red. De este modo, cada valor de cada variable habrá tenido un número de ocurrencias tras generar las muestras. Podemos estimar la probabilidad real de cada valor de cada variable dividiendo el número de ocurrencias que ha tenido en las muestras entre el número total de muestras generadas.

Procedure Forward-Sample (
  $\mathcal{B}$ // Bayesian network over $\mathcal{X}$
)
1 Let $X_1, \ldots, X_n$ be a topological ordering of $\mathcal{X}$
2 **for** $i = 1, \ldots, n$
3 $\boldsymbol{u}_i \leftarrow \boldsymbol{x}\langle \mathrm{Pa}_{X_i} \rangle$ // Assignment to Pa_{X_i} in $x_1, \ldots, x_{i-1}$
4 Sample x_i from $P(X_i \mid \boldsymbol{u}_i)$
5 **return** $(x_1, \ldots, x_n)$

Figura 2.10: Generalización de un algoritmo de muestreo directo [12].

Sin embargo, este algoritmo se encuentra con un problema cuando se hacen consultas del tipo $P(R = T|W = T)$. Una forma de abordar este problema es generar muestras de igual forma, pero ahora se rechazan todas las muestras tales que $W \neq T$. Ahora, el problema es que si $P(W = T)$ es relativamente pequeño, se rechazarán la mayor parte de las muestras generadas.

En la sección 4.1 se desarrolla formalmente y se evalúa programáticamente el algoritmo de "*Probabilistic Logic Sampling*"[14], un ejemplo de este tipo de algoritmos.

2.3.2.2. Likelihood Weighting

Vamos a considerar otra forma de resolver el problema de inferencia aproximada en el que las muestras generadas sean más relevantes para alcanzar una solución; *likelihood weighting* [15].

Consideramos la red de la figura 2.1 y asumimos que tenemos la evidencia de que $R = T$. Si intentamos resolver el problema con el enfoque visto en el anterior algoritmo, obtendremos muestras que en primer lugar tienen en cuenta la probabilidad a priori del nodo C, dando como resultado muestras en las que aproximadamente en un 50 % de los casos $C = T$ y en el 50 % restante $C = F$. Sin embargo, en todas las muestras hemos fijado que $R = T$. Esta evidencia de la que disponemos no ha tenido ninguna repercusión en $P(C)$. Este enfoque no tiene en cuenta que la evidencia introducida aumenta la probabilidad de

que el cielo esté nublado dado que hemos observado que llueve, y nos da una probabilidad por muestreo idéntica a la a priori de la variable.

```
Procedure LW-Sample (
    B,    // Bayesian network over X
    Z = z    // Event in the network
)
1   Let X_1, ..., X_n be a topological ordering of X
2   w ← 1
3   for i = 1, ..., n
4     u_i ← x⟨Pa_{X_i}⟩    // Assignment to Pa_{X_i} in x_1, ..., x_{i-1}
5     if X_i ∉ Z then
6       Sample x_i from P(X_i | u_i)
7     else
8       x_i ← z⟨X_i⟩    // Assignment to X_i in z
9       w ← w · P(x_i | u_i)    // Multiply weight by probability of desired value
10  return (x_1, ..., x_n), w
```

Figura 2.11: Generalización del algoritmo de *Likelihood Weighting* [12].

Mediante el algoritmo de muestreo directo, el nodo R tiene más probabilidad de tomar como valor T cuando su nodo padre C también toma valor T. Consideramos un escenario en el que ejecutamos muestreo con rechazo un número suficientemente grande de veces. Las muestras que generaron $C = T$ hubiesen generado $R = T$ en un 80 % de los casos, mientras que las muestras que generaron $C = F$ solamente generan $R = T$ en un 20 % de los casos. Para conseguir simular este comportamiento a largo plazo en cada muestra, concluimos que una muestra en la que tenemos $C = T$ y evidencia $R = T$ tiene un peso de un 80 % de una muestra, mientras que una en la que tenemos $C = F$ y la misma tiene un 20 % del peso de una muestra. De este modo forzamos que la evidencia tenga repercusión sobre su nodo padre. Aunque las muestras simulan el comportamiento a priori de la variable, el peso asignado a cada muestra construye artificialmente la distribución que debería tener esa variable.

2.3.2.3. Importance Sampling

El *muestreo por importancia* [16] consiste en estimar el valor esperado de una función $f(x)$ de m variables en el dominio $\Omega \subset R^m$ relativa a una distribución $P(X)$, llamada distribución objetivo, tomando X como un conjunto de variables. Se puede estimar el valor esperado de $f(x)$ calculando una serie de muestras $x_1, ..., x_M$ a partir de P, y estimando de la siguiente manera [12]:

$$E_P[f] \approx \frac{1}{M} \sum_{m=1}^{M} f(x_m) \tag{2.5}$$

En ocasiones es preferible utilizar otra distribución Q: es la *distribución de muestreo*. Esto es porque puede ser computacionalmente muy costoso generar muestras a partir de P. P podría ser una distribución a posteriori para la red Bayesiana.

Generalmente, la distribución de muestreo Q puede ser arbitraria. Los únicos requisitos para esta distribución es que $Q(x) > 0$ para todo $P(x) > 0$, y que obtener muestras a partir de ella sea computacionalmente sencillo. De este modo la distribución Q no pasa por alto ningún estado con probabilidad no nula relativa a P. El rendimiento computacional de realizar inferencia aproximada a partir de la distribución Q dependerá en gran medida de lo similar que sea Q a P.

1. Order the nodes according to their topological order.
2. Initialize importance function $\Pr^0(\mathbf{X}\backslash\mathbf{E})$, the desired number of samples m, the updating interval l, and the score arrays for every node.
3. $k \leftarrow 0$, $T \leftarrow \emptyset$
4. **for** $i \leftarrow 1$ **to** m **do**
5. **if** ($i \bmod l == 0$) **then**
6. $k \leftarrow k + 1$
7. Update importance function $\Pr^k(\mathbf{X}\backslash\mathbf{E})$ based on T.

   end if
8. $\mathbf{s}_i \leftarrow$ generate a sample according to $\Pr^k(\mathbf{X}\backslash\mathbf{E})$
9. $T \leftarrow T \cup \{\mathbf{s}_i\}$
10. Calculate Score($\mathbf{s}_i, \Pr(\mathbf{X}\backslash\mathbf{E}, \mathbf{e}), \Pr^k(\mathbf{X}\backslash\mathbf{E})$) and add it to the corresponding entry of every score array according to the instantiated states.

    end for
11. Normalize the score arrays for every node.

Figura 2.12: Generalización del algoritmo de *Importance Sampling* [17].

Al generar muestras a partir de la distribución Q en lugar de P, no podemos estimar el valor de la función $f(x)$ obteniendo la media de las muestras generadas, como en la ecuación 2.5. Este estimador tiene que ser ajustado para compensar el error introducido por la distribución de muestreo incorrecta: Q.

De esta forma el estimador queda de la siguiente manera:

$$E_{P(\mathbf{X})}[f(\mathbf{X})] = E_{Q(\mathbf{X})}\left[f(\mathbf{X})\frac{P(\mathbf{X})}{Q(\mathbf{X})}\right] \tag{2.6}$$

Tras calcular las muestras $D = x_1, ..., x_M$ a partir de $Q(x)$ tenemos el estimador

$$\hat{E}_D(f) = \frac{1}{M}\sum_{m=1}^{M} f(x_m)\frac{P(x_m)}{Q(x_m)}. \tag{2.7}$$

Este estimador es no sesgado. Sintetizando este algoritmo, el problema es estimar la integral

$$V = \int_{\Omega} P(x)dx. \tag{2.8}$$

En el muestreo por importancia [16] esto se resuelve estimando

$$V = \int_{\Omega} \frac{P(x)}{Q(x)} Q(x)dx \tag{2.9}$$

y generando muestras $D = x_1, x_2, ..., x_M$ a partir de Q para obtener el estimador de V:

$$\hat{V} = \frac{1}{M} \sum_{m=1}^{M} \frac{P(x_m)}{Q(x_m)},$$

donde $P(x)$ es la función de densidad de probabilidad de la distribución que queremos alcanzar y $Q(x)$ es la función de densidad de probabilidad de la distribución de muestreo que intenta aproximarse a P. Rubinstein [18] muestra que si $f(X) > 0$, la función $Q(x)$ óptima es

$$Q(X) = \frac{f(X}{V}. \tag{2.10}$$

En este caso, la varianza del estimador es nula. En cualquier caso, si se encuentra una función lo suficientemente cercana a la función óptima se obtienen tasas de convergencia altas.

2.3.2.4. Algoritmo de muestreo de Gibbs

Una de las limitaciones que tienen los métodos vistos hasta ahora es que al insertar evidencia en un nodo o variable afecta al muestreo sólo para nodos descendientes de los mismos. El efecto en nodos no descendientes sólo se tiene en cuenta por los pesos (weights) [12].

Estudiaremos el algoritmo de muestreo de *Gibbs* [12], el ejemplo más sencillo de método de Monte Carlo de cadenas de Markov para realizar inferencia aproximada en Redes Bayesianas. Este algoritmo genera una secuencia de muestras de manera que la primera se obtiene a partir de la distribución de probabilidad a priori, pero las muestras sucesivas que se obtienen se generan a partir de distribuciones que se acercan cada vez más a la distribución a posteriori.

De este modo, nuestro objetivo es, dados ciertos nodos con una evidencia fijada a un determinado valor en una red Bayesiana, estimar la probabilidad de los nodos sin evidencia. A continuación se expone en lenguaje de alto nivel la solución que implementa este algoritmo [19]:

- Inicialmente se genera una muestra de las variables no instanciadas (variables sin evidencia) a partir de cierta distribución [12] que puede ser arbitraria.
- Se asigna a las variables instanciadas los valores correspondientes a la evidencia insertada.

- Para un número grande de iteraciones:
  - Para cada nodo N no instanciado:
    - Calcular la probabilidad de los valores de los nodos N dados los valores actuales de los nodos en su *manto de Markov* (ver 2.2).
    - Asignar a los nodos N el valor de la muestra calculada.
    - Almacenar el valor de estos nodos.

Al terminar este proceso la proporción de iteraciones en las que a un nodo N se le ha asignado un valor concreto se aproxima a la probabilidad a posteriori de ese valor para N.

```
Procedure Gibbs-Sample (
    X     // Set of variables to be sampled
    Φ     // Set of factors defining P_Φ
    P^(0)(X),    // Initial state distribution
    T     // Number of time steps
)
1   Sample x^(0) from P^(0)(X)
2   for t = 1, ..., T
3     x^(t) ← x^(t−1)
4     for each X_i ∈ X
5       Sample x_i^(t) from P_Φ(X_i | x_{−i})
6         // Change X_i in x^(t)
7   return x^(0), ..., x^(T)
```

Figura 2.13: Generalización del algoritmo de *Gibbs Sampling* [12]

Formalmente, una forma de muestrear mediante el método de Gibbs es la siguiente [20]:

Sea el conjunto de variables aleatorias $(X_1, ..., X_{i-1}, X_{i+1}, ..., X_n)$ al que llamaremos X_{-i}, condicionadas por el conjunto de variables instanciadas $e = (e_1, ..., e_m)$. Sea $P(X_1, ..., X_n|e_1, ..., e_m)$ la probabilidad conjunta de las variables $(X_1, ..., X_n)$ condicionadas a e.

1. Inicialización
   a) Se instancia cada X_i a uno de sus posibles valores $x_i, 1 \leq i \leq n$.
   b) Sea $x^{(0)} = (x_1, ..., x_n)$ una muestra obtenida aleatoriamente coherente con la evidencia e.
2. Para $t = 1, 2, ...$
   a) Escoger uniformemente al azar una variable X_i.
   b) Muestrear x_i a partir de $P(X_i|x_{-i}^{t-1}, e)$. Esto es equivalente a muestrear a partir de $P(X_i|MarkovBlanket(X_i))$.
   c) Sea $x^{(t)} = (x_{(-i)}, x_i)$.

Se genera entonces una secuencia de muestras $X^{(0)}, x^{(1)}, ..., x^{(t)}, ...$ que simulan el comportamiento de la distribución conjunta de la red $P(X_1, ..., X_N|e)$.

Ejemplificaremos este funcionamiento basándonos en la red de la figura 2.1. Supongamos que queremos estimar el valor de $P(Rain|Sprinkler = T, WetGrass = T)$. Dado que la evidencia introducida es $Sprinkler = T$ y $WetGrass = T$ se obtendrán muestras de $P(Rain, Cloudy|Sprinkler = T, WetGrass = T)$, de la siguiente forma:

1. Inicialización
   a) Se instancian $Rain = T$ y $Cloudy = T$, por ejemplo
   b) $x^{(0)} = Rain = T, Cloudy = T$
2. Para $t = 1, 2, ...$
   a) Escoger una variable de $Rain, Cloudy$ uniformemente al azar
   b) Si se escoge $Rain$
      1) Se obtiene una muestra de $Rain$ a partir de $P(Rain|Cloudy = [value]_{t-1}, Sprinkler = T, WetGrass = T)$
      2) $x^{(t)} = (Rain = [value]_t, Cloudy = [value]_{t-1})$
   c) Si se escoge $Cloudy$
      1) Se obtiene una muestra de $Cloudy$ a partir de $P(Cloudy|Rain = [value]_{t-1}, Sprinkler = T, WetGrass = T)$
      2) $x^{(t)} = (Rain = [value]_{t-1}, Cloudy = [value]_t)$

Además, necesitamos las probabilidades condicionales de $Rain$ y $Cloudy$ dada la evidencia e, que pueden obtenerse dados los valores de las variables instanciadas utilizando las tablas de probabilidad condicionada y la manta de Markov de la variable en cuestión.

Tomando como ejemplo la red "lluvia"de la figura 2.1, instanciando los nodos *Sprinkler* = *T* y *WetGrass* = *T* y denotando por conveniencia *Cloudy* = *T* como c, *Sprinkler* = *T* como s, *Rain* = *T* como r y *WetGrass* = *T* como w, tenemos que

$$\begin{aligned} P(c|r,s,w) = P(c|s,r) &= \frac{P(c)P(s,r|c)}{P(s,r)} \\ &= \frac{P(c)P(s|c)P(r|c)}{P(c)P(s|c)P(r|c) + P(\neg c)P(s|\neg c)P(r|\neg c)} \\ &= \frac{0.5 \times 0.1 \times 0.8}{0.5 \times 0.1 \times 0.8 + 0.5 \times 0.5 \times 0.2} \\ &= 0.4444 \end{aligned}$$

Se observa directamente que $P(\neg c|r,s,w) = 1 - P(c|r,s,w) = 0.5556$. El resto de probabilidades a posteriori se pueden obtener de forma análoga. De este modo se obtienen las probabilidades condicionales que necesitamos en los pasos $2b1$ y $2c1$.

Capítulo 3

Otros modelos gráficos probabilísticos

En este capítulo se mencionan y desarrollan brevemente otros modelos gráficos probabilísticos, haciendo hincapié en sus diferencias con las redes Bayesianas discretas. Concretamente se desarrollarán modelos basados en grafos no dirigidos y modelos en los que sus variables toman valores continuos, y algunas variantes.

3.1. Modelos gráficos no dirigidos

Estos modelos son útiles para modelar problemas en los que no es trivial asignar una dirección a la interacción entre variables. Estos modelos también se conocen en la literatura como *redes Markovianas* o *redes de Markov*. Hay dos características fundamentales que diferencian las redes de Markov de las redes Bayesianas:

- La forma en que se representan las independencias condicionales: en el caso Markoviano, la interpretación es más simple: las aristas son bidireccionales.
- Cómo se parametriza la distribución de probabilidad conjunta. En el caso de las redes de Markov no es posible utilizar modelos de probabilidad condicionada a otros nodos en la red.

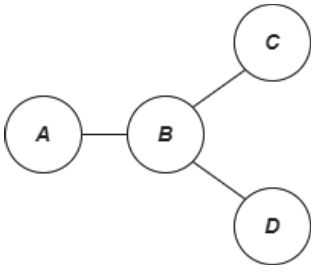
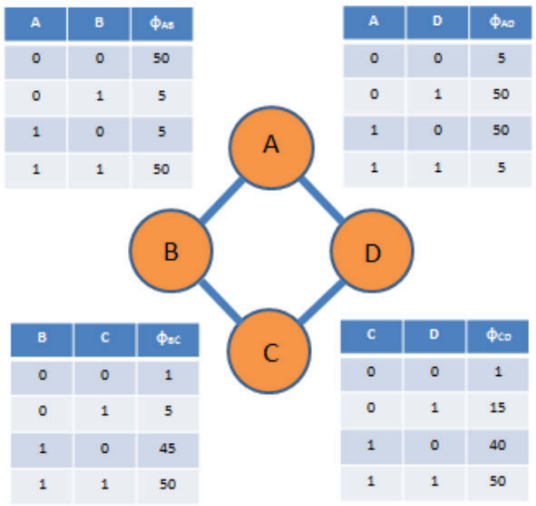


Figura 3.1: Ejemplo de red de Markov. Sus arcos no son dirigidos.

Del gráfico de la figura 3.1 se puede deducir que A, C, D son independientes entre sí si A es conocido.

A la hora de parametrizar en este tipo de modelos se utilizan *factores*, que representan la afinidad entre dos variables [21]. Se puede definir el producto de factores de forma que dados dos factores $\Phi_1(X,Y)$ y $\Phi_2(Y,Z)$ el factor producto $\Psi(X,Y,Z) = \Phi_1(X,Y) \cdot \Phi_2(Y,Z)$. Dados unos factores en una red de Markov se pueden multiplicar para generar la probabilidad conjunta del modelo. Este producto de factores no representa una probabilidad, aúun debe multiplicarse por una función que normaliza la probabilidad.

A	B	C	D	$\phi_{AB}\phi_{BC}\phi_{CD}\phi_{AD}$
0	0	0	0	250
0	0	0	1	37500
0	0	1	0	50000
0	0	1	1	625000
0	1	0	0	1125
0	1	0	1	168750
0	1	1	0	50000
0	1	1	1	625000
1	0	0	0	250
1	0	0	1	375
1	0	1	0	50000
1	0	1	1	6250
1	1	0	0	112500
1	1	0	1	168750
1	1	1	0	5000000
1	1	1	1	625000



A	B	ϕ_{AB}
0	0	50
0	1	5
1	0	5
1	1	50

A	D	ϕ_{AD}
0	0	5
0	1	50
1	0	50
1	1	5

B	C	ϕ_{BC}
0	0	1
0	1	5
1	0	45
1	1	50

C	D	ϕ_{CD}
0	0	1
0	1	15
1	0	40
1	1	50

Figura 3.2: Especificación con factores y productos de factores de una red de Markov

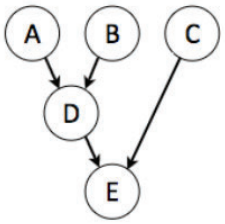
Los modelos gráficos no dirigidos presentan ciertas ventajas y desventajas frente a los dirigidos:

- La especificación de un modelo gráfico no dirigido es más sencilla, pero los factores no tienen una interpretación probabilística directa.
- Determinar independencia entre variables en un modelo no dirigido es más sencillo, en uno dirigido hay que determinar la d-separación.
- Existen propiedades de independencia condicional que no pueden representarse con modelos gráficos dirigidos y viceversa.

3.2. Redes Bayesianas Gaussianas

Hasta ahora todos los modelos gráficos que hemos visto actúan sobre variables discretas. Las redes Bayesianas Gaussianas son redes Bayesianas en las que las variables que la conforman toman valores continuos y sus tablas de probabilidad condicionada son distribuciones Gaussianas lineales.

Una red Bayesiana Gaussiana define una distribución de probabilidad conjun-



$$P(A) = \mathcal{N}(\mu_A, \sigma_A^2)$$
$$P(B) = \mathcal{N}(\mu_B, \sigma_B^2)$$
$$P(C) = \mathcal{N}(\mu_C, \sigma_C^2)$$

$$P(D|A,B) = \mathcal{N}(\beta_{D,0} + \beta_{D,1}A + \beta_{D,2}B, \sigma_D^2)$$
$$P(E|C,D) = \mathcal{N}(\beta_{E,0} + \beta_{E,1}C + \beta_{E,2}D, \sigma_E^2)$$

Figura 3.3: Definición de una red Bayesiana Gaussiana. Las tablas de probabilidad condicionada son distribuciones Gaussianas.

ta que es también Gaussiana [12]. Además, las redes Bayesianas Gaussianas pueden representarse de forma compacta como su distribución de probabilidad conjunta, que se puede calcular de forma simple y eficiente computacionalmente.

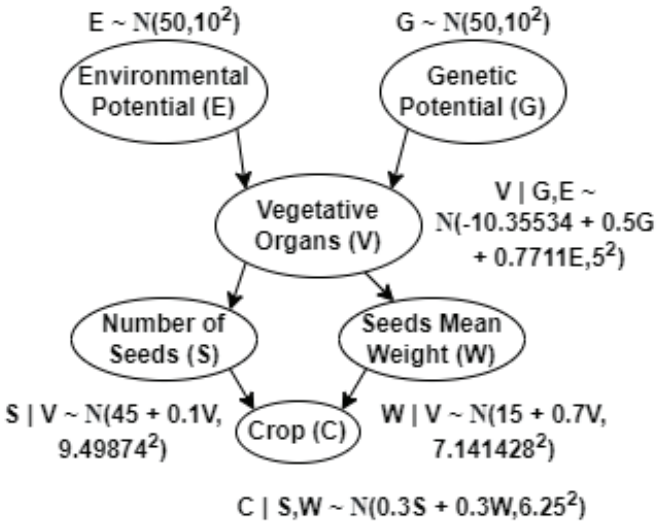


Figura 3.4: [22] Ejemplo de una red Bayesiana Gaussiana de 5 nodos que modeliza el resultado del cultivo en función de otras variables.

Las redes Bayesianas Gaussianas cumplen los siguientes supuestos:

- Cada nodo sigue una distribución normal.
- Los nodos sin padres (nodos raíz) se describen por sus respectivas distribuciones marginales.
- El condicionamiento de los nodos padre viene dado por un término lineal aditivo en la media de las distribuciones de los nodos hijos [22], y no tiene efecto en su varianza. Es decir, cada nodo tiene una varianza específica que no depende del valor de sus padres.

La figura 3.4 muestra un ejemplo de red Bayesiana Gaussiana con su distribución de probabilidad condicionada.

Cabe mencionar que existen modelos mixtos que incorporan variables discretas y continuas. Desarrollar estos modelos supone definir dos casos distintos: una variable continua con padres continuos y discretos, y una variable discreta con padres continuos y discretos.

Consideramos el caso de un nodo X que toma valores continuos. La forma de abordar este caso es ignorar los padres discretos de X, de modo que representamos su probabilidad condicionada como una Gaussiana lineal con términos aditivos en la media de su distribución que dependen de sus padres continuos. Para tener en cuenta a sus padres discretos, se define una Gaussiana lineal con un conjunto de parámetros distinto para cada valor de los padres discretos. Formalmente:

Sea X una variable continua, y sea $\mathbf{U} = U_1, ..., U_m$ el conjunto de sus padres discretos e $\mathbf{Y} = Y_1, ..., Y_k$ el conjunto de sus padres continuos. X tiene una distribución de probabilidad condicional Gaussiana lineal [12] si para cada valor $\mathbf{u} \in \mathbf{U}$, tenemos un conjunto de $k+1$ coeficientes $a_{\mathbf{u},0}, ..., a_{\mathbf{u},k}$ y una varianza $\sigma^2_{\mathbf{u}}$ tal que

$$p(X|\mathbf{u}, \mathbf{y}) = \mathcal{N}\left(a_{\mathbf{u},0} + \sum_{i=1}^{k} a_{\mathbf{u},i} y_i; \sigma^2_{\mathbf{u}}\right)$$

Respecto al otro caso, este modelo no admite que existan variables continuas con hijos que toman valores discretos.

Capítulo 4

Algoritmos

En este capítulo se desarrollará la teoría necesaria para implementar algunos algoritmos que utilizan un mecanismo de inferencia mediante simulación probabilística. Se hará uso de la librería *bnlearn* del entorno de programación R para implementar estos algoritmos y se proveerá un estudio de su rendimiento.

Bnlearn es una librería de R que incluye varios algoritmos para aprender la estructura y realizar inferencia sobre redes Bayesianas discretas y continuas. Además, permite mostrar gráficamente la estructura de redes Bayesianas.

4.1. Probabilistic Logic Sampling

Este algoritmo emplea un mecanismo de simulación estocástica. Mediante este algoritmo se representa una Red Bayesiana mediante una muestra finita de sus posibles escenarios o estados.

Supongamos la representación de una Red Bayesiana por una muestra *M* de *m* escenarios , $M = \{1, 2, ..., m\}$. Supongamos $L_M(x)$ como el valor de una variable x en un escenario M. Podemos representar la incertidumbre de x mediante una "*muestra lógica*", que es un vector de valores para la muestra de escenarios:

$$L(x) = [L_1(x), L_2(x), ..., L_m(x)]$$

Si conocemos la probabilidad a priori $p(x)$, podemos utilizar un generador de números aleatorios para generar una muestra lógica para el evento x [14].

También puede realizarse esta operación a la inversa; dada una muestra lógica $L(x)$ se puede estimar la probabilidad del evento x de esta manera:

$$p(x) = \sum_{M=1}^{m} L_M(x)/m$$

Es decir, la proporción de escenarios en los que x es verdadero.

Para cada distribución de probabilidad condicionada $p(X|Y)$ se genera una muestra lógica para cada una de sus variables independientes con sus probabilidades

correspondientes: $p(x|y)$ y $p(x|\overline{y})$. Los valores de estas muestras lógicas condicionadas $L(x|y)$ y $L(x|\overline{y})$ para cierto escenario M nos dan el valor de x para cualquier estado del nodo padre y [14].

En la figura 2.1 tenemos una Red Bayesiana en la que se muestra cada distribución de probabilidad condicionada en forma tabular. Un escenario específico podría ser $(C = T, S = F, R = T, W = T)$, o en otra notación $(1, 0, 1, 1)$ según una ordenación topológica de los nodos $C \geq S \geq R \geq W$. Este escenario es un ejemplo de muestra lógica posible.

4.1.1. Implementación

Utilizaremos la librería *bnlearn*, en el entorno de programación *R*. Realizaremos las pruebas sobre la red *Rain* de la figura 2.1

Especificamos la estructura de la red y las distribuciones de probabilidad condicionada (ver anexo A.1).

Podemos obtener muestras a partir de la función *cpdist()*. A continuación se muestra el resultado de 5000 muestras (ver anexo A.2).

```
 Cloudy      Rain      Sprinkler  WetGrass
 yes:2435   yes:2464   yes:1538   yes:3195
 no :2565   no :2536   no :3462   no :1805
```

Podemos obtener una tabla de probabilidades conjuntas calculadas a partir de *Probabilistic Logic Sampling* para cada caso posible en la red. La salida es la siguiente (ver anexo A.2):

```
, , Sprinkler = yes, WetGrass = yes

      Rain
Cloudy    yes     no
   yes 0.0404 0.0094
   no  0.0540 0.1830

, , Sprinkler = no, WetGrass = yes

      Rain
Cloudy    yes     no
   yes 0.3086 0.0000
   no  0.0436 0.0000

, , Sprinkler = yes, WetGrass = no

      Rain
Cloudy    yes     no
   yes 0.0002 0.0016
   no  0.0004 0.0186

, , Sprinkler = no, WetGrass = no
```

```
      Rain
Cloudy    yes     no
   yes 0.0420 0.0848
   no  0.0036 0.2098
```

Donde, por ejemplo, $P(Cloudy = yes, Rain = yes, Sprinkler = no, WetGrass = yes) = 0.3086$ y $P(Cloudy = yes, Rain = no, Sprinkler = no, WetGrass = no) = 0.2098$.

Podemos obtener también la probabilidad marginal de una variable especificando otras columnas de la tabla de muestras o utilizando la función *cpquery()* (ver anexo A.2):.

```
[1] 0.6414
```

4.1.2. Evaluación

Para evaluar el rendimiento del código utilizado, tendremos en cuenta el tiempo de ejecución de esta implementación con 5000 muestras. Se evalúa el tiempo de ejecución con el código del anexo A.3.

El tiempo medio de ejecución de este código ha sido de 3.447e-3 segundos. Dado que podemos realizar inferencia exacta [23], podemos tomar una medida del error del algoritmo. Para ello obtenemos una tabla de probabilidad conjunta de la red realizando inferencia exacta con el paquete *gRain* [23], que utiliza el algoritmo de *Junction Tree* [24] (ver anexo A.4).

```
, , Sprinkler = yes, WetGrass = yes

      Rain
Cloudy    yes    no
   yes 0.0396 0.009
   no  0.0495 0.180

, , Sprinkler = no, WetGrass = yes

      Rain
Cloudy   yes no
   yes 0.324  0
   no  0.045  0

, , Sprinkler = yes, WetGrass = no

      Rain
Cloudy   yes    no
   yes 4e-04 0.001
   no  5e-04 0.020

, , Sprinkler = no, WetGrass = no
```

```
      Rain
Cloudy   yes   no
   yes 0.036 0.09
   no  0.005 0.20
```

Podemos obtener la misma estructura de tabla conteniendo el error absoluto para cada probabilidad conjunta.

```
, , Sprinkler = yes, WetGrass = yes

      Rain
Cloudy          yes           no
   yes 2.800000e-03 8.000000e-04
   no  2.500000e-03 1.200000e-03

, , Sprinkler = no, WetGrass = yes

      Rain
Cloudy          yes           no
   yes 3.800000e-03 0.000000e+00
   no  3.600000e-03 0.000000e+00

, , Sprinkler = yes, WetGrass = no

      Rain
Cloudy          yes           no
   yes 5.421011e-20 4.000000e-04
   no  1.000000e-04 8.000000e-04

, , Sprinkler = no, WetGrass = no

      Rain
Cloudy          yes           no
   yes 1.800000e-03 1.600000e-03
   no  2.200000e-03 6.800000e-03
```

Tomaremos como medida del error la raíz del *ECM* o error cuadrático medio. Para un vector de predicciones $\hat{Y}$ y un vector de valores teóricos Y la estimación del *ECM* es:

$$ECM = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2$$

El valor de $\sqrt{ECM}$ está en las mismas unidades que la cantidad que se estima. Esto es una proporción probabilística $\in [0, 1]$. El error obtenido para 5000 muestras en la red Rain es de (ver anexo A.5):

```
[1] 0.002515701
```

Si muestreamos con 15000 muestras en lugar de 5000 obtenemos un $\sqrt{ECM}$ de 1.139e-3, con un tiempo de ejecución medio de 1.164e-2 segundos. Ahora

introducimos cierta evidencia en la red ($Sprinkler = T$) y evaluamos su rendimiento. Es importante que, al realizar inferencia exacta se introduzca también la evidencia para poder comparar correctamente los datos teóricos y simulados. El $\sqrt{ECM}$ obtenido es de (ver Anexo A.5):

```
[1] 0.006816671
```

Obteniendo 5000 muestras con esta evidencia insertada tenemos un $\sqrt{ECM}$ de 6.817e-3 y un tiempo medio de ejecución de 2.64e-3 segundos. Análogamente, con 15000 muestras el $\sqrt{ECM}$ obtenido es de 4.438e-3, con un tiempo medio de ejecución de 8.686e-3 segundos.

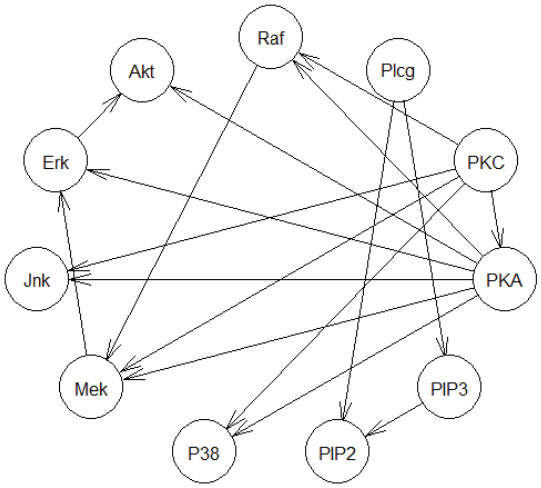


Figura 4.1: Red Bayesiana *SACHS* [25] del repositorio de bnlearn [22], con 11 nodos y 17 arcos.

Vamos a operar ahora sobre una red un poco más compleja; *SACHS* de la figura 4.1.

```
[1] 0.0001794312
```

Obtenemos 5000 muestras (ver anexo A.6) y resulta un $\sqrt{ECM}$ de 1.794e-4, con un tiempo de ejecución medio de 8.142e-3. Simulamos con 15000 muestras y obtenemos un $\sqrt{ECM}$ de 1.777e-4, y un tiempo de ejecución medio de 2.562e-2. Este tiempo de ejecución solo mide la parte referente a obtener las muestras mediante logic sampling.

Introduciendo cierta evidencia en la red ($Mek = HIGH$), y simulando con 5000 muestras, tan sólo 540 cumplen la condición de la evidencia. El resto son rechazadas. Se obtiene un $\sqrt{ECM}$ de 7.097e-4 y un tiempo de ejecución medio de 3.764e-3 segundos. Análogamente, para 15000 muestras se obtiene un $\sqrt{ECM}$ de 7.105e-4, prácticamente el mismo. En este caso el tiempo de ejecución medio es de 9.561e-3 segundos.

4.2. Likelihood weighting

El algoritmo de *Likelihood weighting* [26] es una pequeña variación de *Probabilistic Logic Sampling*. Una muestra en el algoritmo de *Logic Sampling* se rechaza si al generarse, la muestra se contradice con la evidencia introducida en la red. De esta manera, si se introduce evidencia sobre una variable cuya probabilidad marginal es baja, se rechazará una proporción elevada de las muestras.

Para evitar este rechazo de muestras, este algoritmo propone asignar un peso a cada muestra calculada y calcular la media ponderada de las muestras. Este peso refleja la probabilidad de que una muestra no sea rechazada. Este peso se actualiza de modo que por cada variable instanciada (con evidencia insertada), el peso de la muestra se multiplica por la probabilidad de esa evidencia condicionada a sus nodos padres.

Dado que este algoritmo es una pequeña variación del anterior que tiene en cuenta la evidencia introducida, sólo evaluamos para casos con variables instanciadas.

4.2.1. Implementación

En primer lugar obtenemos muestras mediante *Likelihood Weighting* insertando evidencia en la red (ver anexo A.7). Esta evidencia será la misma que hemos introducido para evaluar el anterior algoritmo ($Sprinkler = T$ y $Mek = HIGH$ respectivamente en las redes *Rain* y *SACHS*), de modo que podamos comparar su rendimiento.

```
, , Sprinkler = yes, WetGrass = yes

      Rain
Cloudy    yes     no
   yes 0.3936 0.0934
   no  0.1012 0.3576

, , Sprinkler = no, WetGrass = yes

      Rain
Cloudy    yes     no
   yes 0.0000 0.0000
   no  0.0000 0.0000

, , Sprinkler = yes, WetGrass = no

      Rain
Cloudy    yes     no
   yes 0.0042 0.0076
   no  0.0006 0.0418

, , Sprinkler = no, WetGrass = no

      Rain
```

```
Cloudy    yes     no
   yes 0.0000 0.0000
   no  0.0000 0.0000
```

Como vemos, al condicionar $Sprinkler = T$ la probabilidad marginal de $Sprinkler = F$ es 0. Esta tabla de probabilidades representa las muestras sin tener en cuenta los pesos de cada una. Tendremos en cuenta los pesos en la sección 4.2.2 (evaluación). Además, obteniendo el tamaño de *samples.rain* vemos que están las 5000 muestras generadas, es decir, no se ha rechazado ninguna.

4.2.2. Evaluación

Para evaluar adecuadamente el rendimiento del algoritmo realizamos inferencia exacta introduciendo la misma evidencia para comparar ambos modelos (ver anexo A.8).

```
, , WetGrass = yes

      Rain
Cloudy   yes   no
   yes 0.132 0.03
   no  0.165 0.60

, , WetGrass = no

      Rain
Cloudy         yes          no
   yes 0.001333333 0.003333333
   no  0.001666667 0.066666667
```

Ahora obtenemos la distribución de probabilidad conjunta teniendo en cuenta las muestras y sus pesos.

```
, , Sprinkler = yes, WetGrass = yes

      Rain
Cloudy          yes           no
   yes 0.1328816743 0.0314596190
   no  0.1707137108 0.5963241213

, , Sprinkler = no, WetGrass = yes

      Rain
Cloudy          yes           no
   yes 0.0000000000 0.0000000000
   no  0.0000000000 0.0000000000

, , Sprinkler = yes, WetGrass = no

      Rain
```

```
Cloudy          yes           no
   yes 0.0008720150 0.0040246847
   no  0.0006707808 0.0630533942

, , Sprinkler = no, WetGrass = no

      Rain
Cloudy          yes           no
   yes 0.0000000000 0.0000000000
   no  0.0000000000 0.0000000000
```

Esta tabla es la distribución de probabilidad conjunta simulada mediante este algoritmo. Tiene en cuenta las muestras generadas y el peso de cada una de ellas. Calculamos su $\sqrt{ECM}$ (ver anexo A.8).

```
[1] 0.002824109
```

Con 5000 muestras con la evidencia insertada y utilizando *Likelihood weighting* se obtiene un $\sqrt{ECM}$ de 2.824e-3. Para el mismo caso, el anterior algoritmo obtiene un $\sqrt{ECM}$ de 6.817e-3. Dado que $\sqrt{ECM}$ tiene las mismas unidades que la cantidad que estima, podemos afirmar que este algoritmo tiene un 71 % menos de error que el anterior con 5000 muestras. Análogamente, con 15000 muestras obtenemos un $\sqrt{ECM}$ de 1.892e-3, contra 4.438e-3 que obteníamos con el anterior algoritmo y mismo número de muestras, es decir, un 70 % menos de error. En cuanto a tiempos de ejecución, obtenemos 1.548e-3 y 2.497e-3 respectivamente para 5000 y 15000 muestras, contra los 2.64e-3 y 8.686e-3 del anterior algoritmo.

Dado que si no existe evidencia insertada en la red este algoritmo se comporta de forma idéntica que el visto anteriormente en la sección 4.1, no obtendremos nuevos datos para este caso, ya que sigue la misma distribución tanto en tiempo de ejecución como en $\sqrt{ECM}$ que el anterior.

Los resultados obtenidos para *Likelihood Weighting* son mejores que los de *Probabilistic Logic Sampling* tanto en tiempos de ejecución como en el error (ECM) que produce para la red *Rain*. A continuación probamos para la red *SACHS*.

Con 5000 muestras se obtiene un $\sqrt{ECM}$ de 7.268e-4, y un tiempo de ejecución medio de 2.188e-3 segundos. Ejecutando con 15000 muestras se obtiene un $\sqrt{ECM}$ de 7.115e-4, algo menor que para 5000 muestras. El tiempo de ejecución medio en este caso es de 5.465e-3.

Los resultados obtenidos para esta red nos ofrecen un rendimiento parecido que el algoritmo de *Probabilistic Logic Sampling* en términos de error, sin embargo los tiempos de ejecución son considerablemente menores para el algoritmo de *Likelihood Weighting*.

4.3. Resultados

Con el propósito de sintetizar y comparar de la forma más sencilla posible los resultados obtenidos por estos algoritmos se presenta la información calculada en forma de tablas.

Como mencionamos anteriormente, ambos algoritmos se comportan de forma idéntica si no existe evidencia insertada en la red, de modo que en este caso la distribución de tiempos de ejecución y de error es la misma. Por esta razón no se comparan en las tablas.

A continuación se presenta una comparativa de tiempos de ejecución para cada uno de los algoritmos muestreando con 5000 y 15000 muestras respectivamente.

		Algoritmo	
		PLS	LW
Red	Rain	3.447e-3	-
	Rain (Sprinkler=T)	2.64e-3	1.548e-3
	SACHS	8.142e-3	-
	SACHS (Mek=High)	3.764e-3	2.188e-3

Cuadro 4.1: Tiempo de ejecución en segundos con 5000 muestras

		Algoritmo	
		PLS	LW
Red	Rain	1.164e-2	-
	Rain (Sprinkler=T)	8.686e-3	2.497e-3
	SACHS	2.562e-2	-
	SACHS (Mek=High)	9.561e-3	5.465e-3

Cuadro 4.2: Tiempo de ejecución en segundos con 15000 muestras

Como puede observarse en las tablas, los tiempos de ejecución de *Likelihood Weighting* son considerablemente menores que los obtenidos mediante *Probabilistic Logic Sampling* para todos los casos.

		Algoritmo	
		PLS	LW
Red	Rain	2.516e-3	-
	Rain (Sprinkler=T)	6.817e-3	2.824e-3
	SACHS	1.794e-4	-
	SACHS (Mek=High)	7.097e-4	7.268e-4

Cuadro 4.3: ECM con 5000 muestras

Comparando los datos en términos de error, el $\sqrt{ECM}$ es considerablemente más bajo en el caso de *Likelihood Weighting* para el caso de la red Rain. Sin embargo, los resultados obtenidos para la red SACHS con la evidencia insertada son muy parecidos. En términos netos, *Likelihood Weighting* ofrece mejores resultados que *Probabilistic Logic Sampling*.

		Algoritmo	
		PLS	LW
Red	Rain	1.139e-3	-
	Rain (Sprinkler=T)	4.438e-3	1.892e-3
	SACHS	1.777e-4	-
	SACHS (Mek=High)	7.105e-4	7.115e-4

Cuadro 4.4: ECM con 15000 muestras

Capítulo 5

Conclusiones

En este capítulo se evalúa la consecución de los objetivos planteados para el desarrollo del trabajo en febrero de 2022, así como se proponen algunos objetivos de cara a retomar la investigación en este campo. Se ofrecerán algunas conclusiones personales mencionando algunas dificultades encontradas, y finalmente se hará una breve reflexión con el alineamiento del trabajo con los objetivos de desarrollo sostenible

5.1. Evaluación de objetivos

A continuación se enumeran los objetivos del trabajo evaluando su consecución.

- *Estudiar la formulación de la distribución conjunta de un vector de variables aleatorias factorizada mediante un modelo de red Bayesiana.*

  Tal y como hemos visto al comienzo del capítulo 2, a lo largo del desarrollo de los algoritmos en las secciones 4.1.1 y 4.2.1, así como en la bibliografía, se ha cumplido este objetivo.

- *Mostrar una visión general de los métodos, técnicas, y herramientas de análisis que se van a usar.*

  Como hemos visto en la introducción y en el capítulo de *Algoritmos* (4), las técnicas, métodos y herramientas de análisis se han desarrollado apropiadamente.

- *Estudiar la regla de Bayes simple y multivariante con variables discretas.*

  En la sección 2.3.1 se estudia el teorema de Bayes como preludio a la eliminación de variables como método para realizar inferencia exacta.

- *Considerar varios ejemplos y determinar distribuciones marginales y a priori*

  En la secciones 4.1.1 y 4.2.1 del capítulo de Algoritmos, se obtienen distribuciones marginales y conjuntas a partir de la distribución condicionada para varios casos.

Concluimos que los objetivos finales del trabajo se han alcanzado.

5.2. Líneas futuras

En esta sección se mencionan algunas líneas futuras de desarrollo y propuestas de mejora, así como ideas para aumentar el alcance del proyecto.

- Implementar y evaluar otros algoritmos de inferencia como los que se han descrito en la sección 2.3.
- Implementar y evaluar los algoritmos desarrollados en el capítulo 4 sobre redes más complejas con más nodos y arcos.
- Estudio y uso de otras técnicas de evaluación para los algoritmos implementados, como aquellas basadas en la varianza.
- Estudiar y desarrollar el estado del arte en redes Bayesianas Gaussianas, así como mecanismos de inferencia en las mismas.
- Estudiar y desarrollar otros modelos gráficos probabilísticos, como no paramétricos, semiparamétricos o modelos temporales.
- Estudiar e implementar mecanismos de aprendizaje en redes Bayesianas, como estimación de parámetros o aprendizaje en su estructura.

5.3. Impacto ODS

En esta sección se realiza una breve reflexión sobre cómo se alinea este trabajo con los *Objetivos de Desarrollo Sostenible* (ODS) [27].

- ***Objetivo 3:*** *Garantizar una vida sana y promover el bienestar para todos en todas las edades*

  Las redes Bayesianas, como se ha mencionado anteriormente, constituyen una herramienta que permite modelar procesos caracterizados por la incertidumbre, propio de infinidad de problemas reales [28]. Una de sus primeras aplicaciones prácticas fue en el campo de la medicina como herramienta de diagnóstico de enfermedades. Actualmente, se sigue utilizando para detectar ciertas patologías que pueden ser incluso genéticas, utilizando como variables la presencia de ciertos marcadores como pueden ser algunas proteínas.

  Desde este punto de vista, este trabajo se puede enmarcar como una iniciación a la inferencia en este tipo de modelos, que se siguen utilizando en este campo para facilitar la prevención y detección temprana de algunas enfermedades, acercándonos a garantizar una vida saludable a toda la población.

5.4. Valoración personal

Durante el desarrollo del trabajo he atravesado diversas etapas. En un principio encontré mucha motivación en el tema desarrollado por mi afición a las matemáticas y la estadística, aunque en el primer mes me encontré atascado con la

literatura referente a redes Bayesianas. Esta literatura, cargada de formalismos matemáticos, al principio me infundía cierto respeto, pero he acabado apreciando aún más esta rama del conocimiento.

Encontré ciertas dificultades en el desarrollo de la programación del proyecto ya que R es un entorno de programación que no había utilizado a la fecha de inicio del proyecto. Pese a las limitaciones en este aspecto, he encontrado muy gratificante resolver y evaluar programáticamente el problema de la inferencia aproximada, resultando aún más tangible cuando conseguí evaluar el rendimiento de cada algoritmo.

En general, el desarrollo del trabajo ha sido una experiencia muy positiva. La materia me ha resultado muy interesante, y no me cabe duda de que seguiré leyendo al respecto de aplicaciones e investigaciones al respecto. Me ha resultado muy útil para aprender, ampliar y afianzar conocimientos sobre estadística, aprender a utilizar el entorno de R, mejorar mi redacción y adquirir destreza manejando bibliografía en gran variedad de formatos, así como asimilar algunos conceptos introducidos a lo largo de mis años como estudiante.

Por supuesto, además de mi estudio y trabajo, una parte importante de este trabajo se la debo al apoyo, dedicación y ayuda de mi tutor.

Por mi parte, el resultado del trabajo ha sido muy satisfactorio, se han cumplido los objetivos y he aprendido una infinidad de cosas. Espero que pueda aportar algo a otra mente curiosa por la estadística y la inteligencia artificial.

Bibliografía

[1] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1988.

[2] S. Wright, "The method of path coefficients," *The annals of mathematical statistics*, vol. 5, no. 3, pp. 161–215, 1934.

[3] J. Pearl, *Causality*, 2nd ed. Cambridge University Press, 2009.

[4] BayesFusion. Bayesian networks. [Online]. Available: https://www.bayesfusion.com/bayesian-networks/

[5] S. Nasini, "Notes on bayesian networks - www-eio.upc.es," May 2020. [Online]. Available: https://www-eio.upc.es/~nasini/Blog/BayesianNetworks.pdf

[6] D. Atienza, "Nonparametric models and bayesian networks. applications to anomaly detection," Ph.D. dissertation, Departamento de Inteligencia Artificial. Escuela Técnica Superior de Ingenieros Informáticos. Universidad Politécnica de Madrid, 2021.

[7] V. Pizurica, *Bayesian Inference based programming using smart agent concept*, Oct 2017. [Online]. Available: https://www.linkedin.com/pulse/waylay-engine-bayesian-inference-based-programming-using-pizurica

[8] K. Murphy, "An introduction to graphical models," *Rap. tech*, vol. 96, pp. 1–19, 2001.

[9] F. V. Jensen and T. D. Nielsen, *Bayesian networks and decision graphs*. Springer, 2007, vol. 2.

[10] J. Pearl, *Probabilistic reasoning in intelligent systems*. Elsevier, 2014, vol. 88, no. 3.

[11] W. Mendenhall, R. Beaver, and B. Beaver, *Introduction to Probability and Statistics*. Cengage Learning, 2012. [Online]. Available: https://books.google.es/books?id=60DFDKKQ9Z4C

[12] D. Koller and N. Friedman, *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[13] J. Huggins, "Complexity of inference in bayesian networks | laboratory for intelligent probabilistic systems," Jan 2013. [Online]. Available: https://lips.cs.princeton.edu/complexity-of-inference-in-bayes-nets/

[14] M. Henrion, "Propagating uncertainty in bayesian networks by probabilistic logic sampling," in *Machine intelligence and pattern recognition*. Elsevier, 1988, vol. 5, pp. 149–163.

[15] D. L. Poole and A. K. Mackworth, *Artificial Intelligence: foundations of computational agents*. Cambridge University Press, 2010.

[16] C. Yuan and M. J. Druzdzel, "Importance sampling algorithms for bayesian networks: Principles and performance," *Mathematical and Computer Modelling*, vol. 43, no. 9, pp. 1189–1207, 2006, optimization and Control for Military Applications. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0895717705005443

[17] J. Cheng, "A generic importance sampling algorithm for bayesian networks," 2000. [Online]. Available: https://www.cs.cmu.edu/afs/cs/project/jair/pub/volume13/cheng00a-html/node4.html

[18] R. Y. Rubinstein and D. P. Kroese, *Simulation and the Monte Carlo method*. John Wiley & Sons, 2016.

[19] T. W. Neller, "An introduction to monte carlo techniques in artificial intelligence - part ii," 2014. [Online]. Available: http://modelai.gettysburg.edu/2017/mc2/index.html

[20] P. Schrater, "Csci 5512: Gibbs sampling for approximate inference in bayesian networks," 2011. [Online]. Available: http://vision.psych.umn.edu/users/schrater/schrater_lab/courses/AI2/gibbs.pdf

[21] F. Barreras. (2017, 2) Redes de markov y crf. [Online]. Available: http://www.alvaroriascos.com/mineriadatos/pgm2.pdf

[22] M. Scutari and J.-B. Denis, *Bayesian networks: With examples in R*. CRC Press, 2021.

[23] S. Højsgaard, "Graphical independence networks with the gRain package for R," *Journal of Statistical Software*, vol. 46, no. 10, pp. 1–26, 2012. [Online]. Available: https://www.jstatsoft.org/v46/i10/

[24] M. Paskin, "The junction tree algorithms," 2003. [Online]. Available: https://ai.stanford.edu/~paskin/gm-short-course/lec3.pdf

[25] K. Sachs, O. Perez, D. Pe'er, D. A. Lauffenburger, and G. P. Nolan, "Causal protein-signaling networks derived from multiparameter single-cell data," *Science*, vol. 308, no. 5721, pp. 523–529, 2005. [Online]. Available: https://www.science.org/doi/abs/10.1126/science.1105809

[26] R. D. Shachter and M. A. Peot, "Simulation approaches to general probabilistic inference on belief networks," in *Uncertainty in Artificial Intelligence*, ser. Machine Intelligence and Pattern Recognition, M. HENRION, R. D. SHACHTER, L. N. KANAL, and J. F. LEMMER, Eds.

North-Holland, 1990, vol. 10, pp. 221–231. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780444887382500245

[27] "Desarrollo sostenible." [Online]. Available: https://www.un.org/sustainabledevelopment/es/

[28] V. D. E. Fuster, "Redes bayesianas y diagnóstico médico. una forma diferente de aprender probabilidades condicionadas," *Modelling in Science Education and Learning*, vol. 12, no. 2, pp. 59–76, 2019.

Anexos

Apéndice A

Código fuente

A.1. Creación de red Rain

Importación de la librería bnlearn y declaración de la red Rain, especificando su estructura y distribución de probabilidad condicionada:

```
> library(bnlearn)
> bl.rain ← model2network('[Cloudy][Sprinkler|Cloudy][Rain|Cloudy][
    WetGrass|Sprinkler:Rain]')

> yn ← c("yes","no")
> C ← array(dimnames = list(Cloudy = yn), dim = 2, c(0.5,0.5))
> S ← array(dimnames = list(Sprinkler = yn, Cloudy = yn), dim = c(2, 2)
    , c(0.1,0.9,0.5,0.5))
> R ← array(dimnames = list(Rain = yn, Cloudy = yn), dim = c(2, 2), c
    (0.8,0.2,0.2,0.8))
> W ← array(dimnames = list(WetGrass = yn, Sprinkler = yn, Rain = yn),
    dim = c(2, 2, 2), c(0.99,0.01,0.9,0.1,0.9,0.1,0,1))
> cpts ← list(Cloudy = C, Sprinkler = S, Rain = R, WetGrass = W)
> bl.rain.fit = custom.fit(bl.rain, cpts)
```

A.2. Muestreo PLS

Muestreo mediante *Probabilistic Logic Sampling* y distribución de probabilidad conjunta obtenida a partir del mismo:

```
> set.seed(0)
> samples.rain ← cpdist(bl.rain.fit, nodes = nodes(bl.rain.fit),
    evidence = TRUE)
> summary(samples.rain)

> t ← table(samples.rain[,c('Cloudy','Rain','Sprinkler','WetGrass')])
> prop.table(t)

> cpquery(bl.rain.fit, event = (WetGrass == "yes"),evidence=TRUE)
```

A.3. Evaluación del tiempo de ejecución

Código utilizado para medir el tiempo de ejecución del muestreo:

```
t0 ← Sys.time()
samples.rain ← cpdist(bl.rain.fit,nodes = nodes(bl.rain.fit),evidence=
    TRUE,n=5000)
t1 ← Sys.time()-t0
print(t1)
```

A.4. Inferencia exacta

Código para realizar inferencia exacta sobre la red y obtener los datos en formato tabular, así como el error absoluto para cada valor de la distribución conjunta:

```
> library(gRain)
> gr.rain←as.grain(bl.rain.fit)
> q ← querygrain(gr.rain, nodes = nodes(bl.rain.fit), type = "joint")
> q

> abs(q-t)
```

A.5. Medida del error en Rain sin instanciar

```
> sqrt(mean((q-t)^2))

> rainEv ← setEvidence(gr.rain, nodes="Sprinkler", states=c("yes"))
> qRainEv ← querygrain(rainEv, nodes = nodes(bl.rain.fit), type = "
    joint")
> samples.rain ← cpdist(bl.rain.fit, nodes=nodes(bl.rain.fit),evidence
    =(Sprinkler == "yes"),n=5000)
> t ← table(samples.rain)
> t ← prop.table(t)
> sqrt(mean((qRainEv-t[,,"yes",])^2))
```

A.6. Medida del error en SACHS

```
> load(url("http://www.bnlearn.com/bnrepository/sachs/sachs.rda"))
> sachs.fit ← bn; rm(bn)
> gr.sachs ← as.grain(sachs.fit)
> q ← querygrain(gr.sachs, nodes = nodes(sachs.fit), type = "joint")
```

```
> samples.sachs ← cpdist(sachs.fit, nodes=nodes(sachs.fit),evidence=
    TRUE,n=5000)
> t ← table(samples.sachs)
> t ← prop.table(t)
> sqrt(mean((q-t)^2))
```

A.7. Muestreo LW

```
> samples.rain ← cpdist(bl.rain.fit, nodes=nodes(bl.rain.fit),evidence=
    list(Sprinkler="yes"),n=5000, method="lw")
> t←table(samples.rain)
> t←prop.table(t)
> t
```

A.8. Medida del error en LW

```
> gr.rain ← as.grain(bl.rain.fit)
> gr.rain ← setEvidence(gr.rain, nodes='Sprinkler',states='yes')
> qEvRain ← querygrain(gr.rain, nodes=nodes(bl.rain.fit),type="joint")
> qEvRain

> w ← attr(samples.rain,'weights')
> t ← prop.table(xtabs(w ~ Cloudy+Rain+Sprinkler+WetGrass, data=cbind(
    samples.rain,w=w)))
> t

> sqrt(mean((qEvRain-t[,,"yes",])^2))
```

Este documento esta firmado por

Firmante	CN=tfgm.fi.upm.es, OU=CCFI, O=ETS Ingenieros Informaticos - UPM, C=ES
Fecha/Hora	Thu Jun 30 22:49:59 CEST 2022
Emisor del Certificado	EMAILADDRESS=camanager@etsiinf.upm.es, CN=CA ETS Ingenieros Informaticos, O=ETS Ingenieros Informaticos - UPM, C=ES
Numero de Serie	561
Metodo	urn:adobe.com:Adobe.PPKLite:adbe.pkcs7.sha1 (Adobe Signature)