UNIVERSIDAD
DE GRANADA

Escuela Técnica Superior de Ingenierías Informática y
Telecomunicación
Facultad de Ciencias

Doble Grado en Ingeniería Informática y Matemáticas

TRABAJO DE FIN DE GRADO

Aplicación de Redes Bayesianas a datos genéticos

Presentado por:
Álvaro Beltrán Camacho

Tutor:
Ofelia Paula Retamero Pascual
Centro de Investigación Tecnologías de Infor. y Comunicaciones

Manuel Gómez Olmedo
Dpto. de Ciencias de la Computación e Inteligencia Artificial

Curso académico 2020-2021

Aplicación de Redes Bayesianas a datos genéticos

Álvaro Beltrán Camacho

Álvaro Beltrán Camacho *Aplicación de Redes Bayesianas a datos genéticos*.
Trabajo de fin de Grado. Curso académico 2020-2021.

Responsable de tutorización

Ofelia Paula Retamero Pascual
Centro de Investigación Tecnologías de Infor. y Comunicaciones

Manuel Gómez Olmedo
Dpto. de Ciencias de la Computación e Inteligencia Artificial

Doble Grado en Ingeniería Informática y Matemáticas

Escuela Técnica Superior de Ingenierías Informática y Telecomunicación
Facultad de Ciencias

Universidad de Granada

Declaración de originalidad

D./Dña. Álvaro Beltrán Camacho

Declaro explícitamente que el trabajo presentado como Trabajo de Fin de Grado (TFG), correspondiente al curso académico 2020-2021, es original, entendido esto, en el sentido de que no se han utilizado para la elaboración del trabajo fuentes sin citarlas debidamente.

En Granada a 11 de julio de 2021

Fdo: Álvaro Beltrán Camacho

Índice general

Índice de figuras XI

Índice de tablas XIII

Resumen XV

Summary XVII

Introducción XXI

0.1. Motivación XXII
0.2. Objetivos XXII
0.3. Estructura XXII

I. Teoría Redes Bayesianas 1

1. Teoría de la Probabilidad 3

1.1. Distribución de probabilidad 3
1.1.1. Espacio muestral 3
1.1.2. Distribución de probabilidad 3
1.2. Conceptos Básicos 4
1.2.1. Probabilidad Condicionada 4
1.2.2. Teorema de Bayes 4
1.3. Variables aleatorias 5
1.3.1. Independencia 6
1.3.2. Distribución Condicionada 6
1.3.3. Independencia Condicional 6
1.3.4. Función de densidad 7
1.4. Distribuciones normales 7
1.4.1. Distribución normal 7
1.4.2. Distribución normal multivariante 7

2. Teoría de Grafos 9

2.1. Grafos, Nodos y Arcos 9
2.2. La estructura de un grafo 10

3. Representación de la Red Bayesiana 13

3.1. Redes Bayesianas 13
3.1.1. Grafos y distribuciones 14
3.1.2. Ejemplo red asia 15
3.2. Independencia en grafos 16
3.2.1. D-separación 16
3.2.2. Robustez y completitud 17

3.2.3. I-equivalencia . . . 18
3.3. Mapas . . . 19
3.3.1. I-mapa minimal . . . 19
3.3.2. I-mapa perfecto . . . 19

4. Modelos Probabilísticos Locales **21**
4.1. Introducción . . . 21
4.2. DPCs en formato tabla . . . 21
4.3. DPCs deterministas . . . 22
4.4. DPCs representados mediante árboles . . . 22
4.5. DPCs definidas mediante reglas . . . 23
4.6. Influencia causal . . . 23
4.6.1. Modelo Noisy-OR . . . 23
4.6.2. Modelos lineales generalizados . . . 24
4.6.2.1. Variables binarias . . . 24
4.6.2.2. Variables multivaluadas . . . 24
4.7. Variables continuas . . . 25
4.8. Modelos híbridos . . . 25

5. Inferencia **27**
5.1. Introducción . . . 27
5.2. Análisis de complejidad . . . 27
5.3. Eliminación de variables . . . 28
5.3.1. Marginalización de factores . . . 28
5.3.2. Algoritmo de eliminación de variables . . . 29

6. Aprendizaje **31**
6.1. Motivación . . . 31
6.2. Formalización de la idea de aprendizaje . . . 31
6.2.1. Estimación de la densidad . . . 31
6.2.2. Predicción . . . 33
6.2.3. Extracción de conocimiento . . . 34
6.3. Optimización . . . 34
6.3.1. Riesgo empírico y sobreajuste . . . 34
6.4. Tareas del aprendizaje . . . 35

7. Estimación del modelo **37**
7.1. Estimación de la estructura . . . 37
7.1.1. Basada en restricciones . . . 37
7.1.1.1. *Grow-Shrink* (GS) . . . 37
7.1.1.2. *Incremental Association Markov Blanket* (IAMB) . . . 39
7.1.2. Aprendizaje basado en métricas . . . 40
7.1.2.1. *Hill-Climbing* (HC) . . . 41
7.1.3. Metodología híbrida . . . 41
7.1.3.1. *Max-Min Hill-Climbing* (MMHC) . . . 41
7.2. Estimación de los parámetros . . . 43
7.2.1. Estadístico máximo verosímil (EMV) . . . 43
7.2.1.1. Descomposición función de verosimilitud . . . 43

7.2.1.2. Tablas DPCs 44
7.2.1.3. Red Bayesiana Gaussiana 45

II. Aplicación a Datos Genéticos **47**

8. Aplicaciones de las Redes Bayesianas al análisis de datos genéticos **49**
8.1. *Using bayesian networks to analyze expression data* 49
8.2. *Using Bayesian networks to discover relations between genes, environment, and disease* 51

9. Base de datos genética **55**
9.1. Empalme alternativo (*splicing*) 55
9.2. Explicación de la base de datos y análisis preliminar 56

10. Estudio experimental en una base de datos genética **61**
10.1. Introducción 61
10.2. Arquitectura del cuaderno 61
10.2.1. Sistema *R* embebido en *Python* 62
10.3. Librerías usadas 62
10.3.1. Librerías *Python* 62
10.3.1.1. *Pandas* 62
10.3.1.2. *Numpy* 63
10.3.1.3. *Matplotlib* 63
10.3.1.4. *Scikit-Learn* 63
10.3.2. Librerías *R* 64
10.3.2.1. *BNlearn* 64
10.3.2.2. *Graphviz* 65
10.4. Cuaderno base de datos genética 65
10.4.1. Carga de datos 65
10.4.2. Limpieza y preprocesamiento de los datos 66
10.4.2.1. Eliminar instancias irrelevantes 66
10.4.2.2. Selección de variables 66
10.4.3. División de conjuntos 68
10.4.4. Aprendizaje estructural 68
10.4.4.1. Comparativa de estructuras 77
10.4.4.2. Elección de la estructura de RB 80
10.4.5. Estimación paramétrica 80
10.4.6. Inferencia y predicción 81
10.4.6.1. Bondad del modelo 81

11. Conclusiones y estudios futuros **83**

A. Primer apéndice: Distribución de probabilidad **85**

Bibliografía **97**

Índice de figuras

2.1. Grafo no dirigido (izquierda), Grafo dirigido (derecha) 9
2.2. Padres, ancestros, vecinos, hijos y descendientes del nodo A $\in V$ 10

3.1. Red Asia 15
3.2. (a) efecto causal indirecto, (b) efecto evidencial indirecto, (c) Causa común, (d) Efecto común 17

4.1. (a) Red de ejemplo. (b) Árbol-DPC para $P(J|C, L_1, L_2)$. 22

5.1. ejemplo de marginalización de factores: sumar B 28

8.1. Red Bayesiana 50
8.2. Red Bayesiana local al gen SVS1 51
8.3. Red Bayesiana intermedia 52
8.4. Red Bayesiana final 53

9.1. Creación ARN 55
9.2. Síntesis de proteínas 56
9.3. Frecuencia de los valores por posición 57
9.4. Frecuencias etiquetas 58
9.5. Diagrama de barras Valores posiciones por Etiqueta 59

10.1. Arquitectura del cuaderno 61
10.2. P-valores 67
10.3. División de conjuntos 68
10.4. Estructura algoritmo HC 73
10.5. Estructura algoritmo MMHC 74
10.6. Estructura algoritmo GS 75
10.7. Estructura algoritmo IAMB 76
10.8. Comparativa HC con MMHC 78
10.9. Comparativa HC con IAMB 79
10.10Comparativa IAMB con MMHC 79
10.11Matriz de confusión sobre el conjunto de comprobación 82

Índice de tablas

4.1. Ejemplo DPC en formato tabla 21

9.1. Posibles valores de las posiciones de la secuencia genética 56
9.2. Ejemplo instancias base de datos 57
9.3. Nº de genes por etiqueta 58
9.4. Frecuencia valores posiciones por etiqueta 58

10.1. Formato datos cargados 65
10.2. Formato tabla de variables 66
10.3. Métricas sobre las estructuras de los algoritmos 80

Resumen

En este trabajo se van a estudiar teóricamente las redes Bayesianas, desde su definición hasta los conceptos básicos asociados. Luego, se analizarán las posibles representaciones de las distribuciones de probabilidad condicionadas que presenta la red Bayesiana. Mediante la distribución y la estructura de la red, se presentará un algoritmo para hacer inferencia. El estudio teórico, acabará presentando la tarea de aprendizaje y el proceso para aprender una red Bayesiana a partir de un conjunto de datos.

Una vez finalizado el estudio teórico, se hace un estudio experimental sobre el aprendizaje de una red Bayesiana a partir de una base de datos genética. La primera tarea consistirá en hacer un análisis previo de los estudios realizados en el campo de las redes Bayesianas con datos genéticos. Luego, se realizará un análisis explicativo de la base de datos elegida para la experimentación. Y finalmente, se realizará un cuaderno *Python* para realizar la experimentación comentada. En este cuaderno, se realiza la carga y limpieza de los datos, además de preprocesamiento y una división de conjuntos para el aprendizaje. Posteriormente, se realiza la estimación de la red Bayesiana a partir de los datos, se hace inferencia y se estudia la bondad del modelo.

PALABRAS CLAVE: red Bayesiana, aprendizaje, distribución, algoritmo, probabilidad, genética, preprocesamiento.

Summary

Bayesian Networks take their name from Bayes' theorem, a theorem proposed in the 18th century by Thomas Bayes. This theorem allows us to determine the conditional probability of one event with respect to another, an essential element for the understanding of probabilistic graphical models.

Over the years, the idea of representing the relationships between the variables of a problem through the use of a graphical structure has become popular. graphical structure. In the area of statistical physics, this idea goes back to Gibbs (1902), who used an undirected graph to represent the distribution over a system of interacting particles. In the area of genetics, this idea goes back to the work on path analysis by Sewal Wright (Wright 1921, 1934). In the field of statistics, the idea of analyzing interactions between variables was first proposed by Bartlett (1935), in the study of contingency tables. In the field of computer science, graphical models came by the hand of Artificial Intelligence. In the late 1980s, the development of the Bayesian network by Judea Pearl and the invention of large-scale expert systems brought Bayesian networks to prominence.

Today, probabilistic graphical models are used in a lot of areas such as psychology, medicine, finance, failure treatment, genetic data analysis, marketing, speech recognition and much more.

In this work we will focus on Bayesian networks, a specific case of probabilistic graphical models. These allow us to cover problems of great complexity, due to the large number of attributes to be studied and their relationships among them. They use a representation based on graphs as a basis for encoding high-dimensional distributions. These graphs are composed of nodes, which correspond to the variables, and directed edges, which simulate the relationships between the variables. On the other hand, by using inference and learning, we will be able to generate an intelligent system capable of reasonably understanding the model. In inference and learning, numerous techniques are applied. In learning we will study the formal idea and how it is transformed into the estimation of a complete Bayesian network model, estimating the structure and the distribution.

The work is divided into two distinct parts, one for the theoretical study of Bayesian networks; and the other for the application to a genetic database. The first part consists of seven chapters and the second part of four.

In the Bayesian network theory part, we begin by introducing basic concepts of probability theory such as probability distribution, Bayes' rule, random variables and normal distributions. Then, in the next chapter, the basic concepts of graph theory are introduced. In this chapter, acyclic directed graphs are defined, a fundamental concept to study the structure of a Bayesian network.

The third chapter, called Bayesian network representation, introduces the concept of Bayesian network and explains a simple example of the Asia network. Subsequently, concepts such as d-separation, I-equivalence and maps are studied. In the next chapter, local probabilistic models, the ways of representing conditional probability distributions on discrete, continuous, binary, multivalued and hybrid variables are explained. To represent the conditional probability distribution, representations in tabular, deterministic, tree, rule, etc. format are used.

In the fifth chapter, inference, an algorithm to perform inference on probabilistic graphical models and specifically on a Bayesian network is studied. The algorithm is called variable elimination and is based on the grouping of distributions until the distribution of the desired variables is obtained. In this process, the concept of factor is defined, which is essential for the algorithm to work. In the following chapter the idea of learning is formalized. For this purpose, learning is reformulated as an optimization problem of a function in a given search space. The space will be the space of Bayesian networks.

The last chapter of the first part, model estimation, begins by explaining the process of estimating the structure of a Bayesian network and ends with the estimation of the parameters of the distribution given that structure. For the estimation of the structure, four algorithms are explained: *Grow-Shrink (GS), Incremental Association Markov Blanket (IAMB), Hill-Climbing (HC) and Max-Min Hill-Climbing (MMHC)*; with these algorithms an acyclic directed graph is learned based on the data. Then, once the structure is defined, the parametric estimation of the relationships given by the structure is presented using the maximum likelihood estimator.

The second part, application to genetic data, begins with a chapter dedicated to review two articles where a Bayesian network is estimated from genetic data. Then, in the next chapter, called genetic database, an explanatory analysis of the database *Molecular Biology (Splice-junction Gene Sequences) Data Set* [G Towell and Shavlik, 1991] is made. This analysis tries to understand the database and to obtain relevant information for the future preprocessing of the database. Also, the classification problem of the database is defined.

Finally, in the tenth chapter, an experimental study on a genetic database. The *Python* notebook for the realization of the learning of a Bayesian network on the data previously discussed is described. The first thing described in the chapter is the notebook architecture. The architecture is based on a *Python* notebook with an *R* embedded system, all developed in the *Google Collaboratory* notebooks. Then, the libraries used in the *Python* and *R* languages are described. Once the general structure of the notebook is known, it is explained. First, the data previously stored in *Google Drive* is loaded. Then, the database is cleaned and a selection of variables is made due to the number of variables. After this, the set is divided into three parts to proceed to the learning of the Bayesian network. The R embedded system is used to use the *BNlearn* library to learn the Bayesian network according to the database. Finally, inference is performed and the goodness of the model is studied.

After having a complete and theoretical vision of Bayesian networks as well as of the learning and inference process, the usefulness of the mathematical model on genetic data is confirmed. The results obtained for the database are very good and it has been possible to represent the existing relationships between the positions of the DNA sequence and the

class. In addition, the libraries present a large number of functions to make the process of learning and inference on the data easier.

In the future, it would be of interest to analyze more typologies of Bayesian networks such as dynamic, seeded or temporal Bayesian networks. Moreover, in the field of genetics, it seems logical to continue investigating new relationships between data because the Bayesian network is a white box model. Finally, it seems interesting to test or analyze from an expert the point of view of the relationships obtained from learning a Bayesian network.

PALABRAS CLAVE: Bayesian network, learning, distribution, algorithm, probability, genetics, preprocessing.

Introducción

Las Redes Bayesianas toman su nombre del teorema de Bayes, teorema propuesto en el siglo XVIII por el Reverendo Thomas Bayes. Este teorema nos permite determinar la probabilidad condicionada de un evento con respecto a otro, elemento primordial para el entendimiento de los modelos gráficos probabilísticos. [Koller and Friedman, 2009]

A lo largo de los años, se fue popularizando la idea de representar las relaciones entre las variables de un problema mediante el uso de una estructura gráfica. En el área de la física estadística, esta idea se remonta a Gibbs (1902), quien utilizó un gráfico no dirigido para representar la distribución sobre un sistema de partículas que interactúan. En el área de la genética, esta idea se remonta al trabajo sobre el análisis de caminos de Sewal Wright (Wright 1921, 1934). En el campo de la estadística, la idea de analizar las interacciones entre las variables fue propuesta por primera vez por Bartlett (1935), en el estudio de las tablas de contingencia. [Koller and Friedman, 2009]

En el campo de la informática, los modelos gráficos llegaron de la mano de la Inteligencia Artificial. Pero se rechazaron los métodos probabilísticos en general a cambio de una serie de formalismos para el razonamiento en condiciones de incertidumbre. Sin embargo, a finales de 1980, los métodos probabilísticos volvieron a ponerse de moda debido a dos factores: el desarrollo de la red Bayesiana por Judea Pearl y la invención de sistemas expertos a gran escala. [Koller and Friedman, 2009]

En la actualidad, los modelos gráficos probabilísticos se usan en una infinidad de áreas como psicología, reflejado en el artículo *Las redes bayesianas como herramientas de modelado en psicología* [Puga et al., 2007]; medicina; finanzas, reflejado en el artículo *Cálculo del valor en riesgo operacional mediante redes bayesianas para una empresa financiera* [Aragón et al., 2016]; tratamiento de fallos; análisis de datos genéticos; comercialización; reconocimiento del habla y mucho más.

En este trabajo nos centraremos en las redes Bayesianas, caso concreto de los modelos gráficos probabilísticos. Estos, nos permiten abarcar problemas de gran complejidad, debido a las grandes cantidades de atributos a estudiar y sus relaciones. Utilizan una representación basada en grafos como base para codificar distribuciones de altas dimensiones. Estos grafos están compuestos por nodos, que corresponden a las variables, y por aristas dirigidas, que representan las relaciones entre las variables [Koller and Friedman, 2009]. Por otro lado, ayudándonos de la inferencia y el aprendizaje, seremos capaces de generar un sistema inteligente capaz de comprender razonablemente nuestro modelo. Tanto en inferencia como en aprendizaje se aplican numerosas técnicas. En este trabajo se estudiará el algoritmo de eliminación de variables para inferencia. Mientras que en aprendizaje se hará un estudio sobre la idea formal y como esta se transforma en la estimación de un modelo completo de red Bayesiana, estimando tanto la estructura como la distribución.

Finalmente, se aplicará todo lo aprendido a una base de datos genética del repositorio UCI [Dua et al., 2017] llamada *Molecular Biology (Splice-junction Gene Sequences) Data Set* [G Towell and Shavlik, 1991]. Esta base de datos recopila secuencias de ADN con 60 posiciones donde se produce el cambio exón-intrón y viceversa, con el objetivo de poder inferir

en base a una secuencia de ADN este cambio.

0.1. Motivación

Tras indagar sobre redes Bayesianas y ver la asombrosa cantidad de aplicaciones nos pareció de gran interés estudiar este concepto que no se imparte en profundidad en ninguna de las asignaturas estudiadas en ambos grados.

Además, nos pareció muy interesante implementar una red Bayesiana para tratar de obtener conclusiones de una base de datos genética. Esta idea se ve motivada por los resultados de los trabajos de investigación *Using Bayesian networks to discover relations between genes, environment, and disease* [Su et al., 2013] y *Using Bayesian Networks to Analyze Expression Data* [Friedman et al., 2000], donde aprenden una red Bayesiana a través de una base de datos genética con el objetivo de hacer inferencia sobre ella.

0.2. Objetivos

Los objetivos de este trabajo se van a basar en el estudio teórico de las redes Bayesianas y su posterior aplicación a un caso experimental. En el estudio teórico, se busca aprender la teoría referente a redes Bayesianas; a la representación de las distribuciones de las mismas; a la inferencia sobre las redes Bayesianas; y , al aprendizaje y estimación de las redes Bayesianas a partir de un conjunto de datos dado.

En el estudio experimental se busca construir de forma automática una red Bayesiana sobre la base de datos genética *Molecular Biology (Splice-junction Gene Sequences) Data Set* [G Towell and Shavlik, 1991], haciendo un estudio previo sobre trabajos involucrados en esta tarea y finalizando con la creación de un cuaderno *Python* con la experimentación comentada.

0.3. Estructura

El trabajo consta de dos partes. La primera parte hace referencia al estudio teórico de las redes Bayesianas y la segunda a la aplicación a la base de datos genética.

La primera parte, llamada teoría de redes Bayesianas, consta de siete capítulos. En el primer capítulo [1], llamado teoría de la probabilidad, se definen conceptos básicos relativos a la probabilidad. En el segundo capítulo [2], teoría de grafos, se explican los conceptos necesarios sobre grafos. En el tercer capítulo [3], representación de la red Bayesiana, se definen la red Bayesiana y algunos conceptos de la misma. En el cuarto capítulo [4], modelos probabilísticos locales, se estudian las representaciones de las distribuciones condicionales de probabilidad de la red Bayesiana. En el quinto capítulo [5], inferencia, se explica un algoritmo para hacer inferencia sobre una red Bayesiana. En el sexto capítulo [6], aprendizaje, se formalizan las ideas de aprendizaje sobre los modelos gráficos probabilísticos. En el séptimo y último capítulo [7] de la primera parte, se hace un estudio teórico sobre la estimación de una red Bayesiana.

La segunda parte, aplicación a datos genéticos, consta de tres capítulos. El primer capítulo [8], aplicaciones de las redes Bayesianas al análisis de datos genéticos, se analizan dos ar-

tículos sobre redes Bayesianas en datos genéticos. En el segundo capítulo [9], base de datos genética, se explica y analiza en profundidad la base de datos que se va a usar en la experimentación. En el tercer capítulo [10], estudio experimental en una base de datos genética, se explica el desarrollo del cuaderno *Python* creado para el aprendizaje de una red Bayesiana sobre la base de datos genética. Finalmente, se presenta un capítulo [11] sobre conclusiones y posibles trabajos futuros.

Parte I.

Teoría Redes Bayesianas

En esta primera parte se va a realizar el estudio teórico de las redes Bayesianas, haciendo un breve estudio inicial de teoría de la probabilidad en el capítulo 1 para luego continuar con teoría de grafos en el capítulo 2. Posteriormente, nos introduciremos en la teoría de redes Bayesianas con la representación de las mismas en el capítulo 3, el estudio de los modelos probabilísticos locales en el capítulo 4, la inferencia en redes Bayesianas en el capítulo 5, la formalización de la idea de aprendizaje en el capítulo 6 y finalmente la estimación del modelo mediante un conjunto de datos en el capítulo 7.

1. Teoría de la Probabilidad

En este capítulo se va a estudiar qué es un espacio de probabilidad, cuáles son sus propiedades básicas y qué es una distribución de probabilidad. Para luego, introducir las variables aleatorias; estudio que será de vital importancia en la teoría de redes Bayesianas.

1.1. Distribución de probabilidad

Cuando se habla coloquialmente de probabilidad; el término hace referencia al grado de confianza de que un cierto evento ocurra en un marco de incertidumbre. Esta definición; a priori, poco matemática, nos da una idea de lo que es la probabilidad.

La distribución de probabilidad es la asignación de un valor probabilístico a un evento dado. Estos eventos pueden tener naturaleza discreta o continua. Para comenzar en la teoría de la probabilidad se va a definir el espacio muestral.

1.1.1. Espacio muestral

El **espacio muestral** es el conjunto de todos los posibles valores que puede tomar un determinado evento, se denota por Ω. Además, se denota por $\mathcal{S}$ al conjunto de los eventos medibles. Formalmente, se tiene que $\alpha \in \mathcal{S}$ es un subconjunto de Ω. Para que la teoría de la Probabilidad tenga sentido, se necesita que el espacio muestral cumpla tres axiomas:

- $\emptyset$ (conjunto vacío) , Ω (conjunto trivial) $\in \mathcal{S}$
- Ω es cerrado para la unión; es decir, si $\alpha, \beta \in \mathcal{S} \Rightarrow \alpha \cap \beta \in \mathcal{S}$
- Ω es cerrado para el complementario; es decir, si $\alpha \in \mathcal{S} \Rightarrow \Omega - \alpha \in \mathcal{S}$

Una vez se ha definido el espacio muestral. Se necesita definir la probabilidad de que ocurra un determinado evento, con lo que aparece el concepto de distribución de probabilidad.

1.1.2. Distribución de probabilidad

Definición 1.1. La **distribución de probabilidad** P sobre $(\Omega, \mathcal{S})$ es la función que asigna la probabilidad a un determinado evento de $\mathcal{S}$. Estos valores deben cumplir:

- $P(\alpha) \geq 0 \ \forall \alpha \in \mathcal{S}$.
- $P(\Omega) = 1$.
- Para $\alpha, \beta \in \mathcal{S}$ con $P(\alpha \cap \beta) = 0 \Rightarrow P(\alpha \cup \beta) = P(\alpha) + P(\beta)$

1.2. Conceptos Básicos

Ahora que se ha definido la distribución de probabilidad, es necesario definir dos conceptos básicos como son la probabilidad condicionada y la regla de Bayes.

1.2.1. Probabilidad Condicionada

Se quiere definir el significado de la probabilidad de que ocurra un evento condicionado a que ocurra otro. Además, es necesario definir el significado de que dos eventos son independientes; es decir, que la probabilidad de que se dé un evento no influye en la probabilidad de que ocurra otro evento. Esta idea se formaliza usando definiciones de los libros *Probabilistic Graphical Models* [Koller and Friedman, 2009] y *Probability Theory* [Borovkov, 2013].

Definición 1.2. Sea $(\Omega, \mathcal{S}, P)$ un espacio de probabilidad y α , $\beta \in \mathcal{S}$. Si $P(\beta) > 0$, la probabilidad de α sabiendo que ha ocurrido β se denota como $P(\alpha|\beta)$, y se define como: $P(\alpha|\beta) := \frac{P(\alpha \cap \beta)}{P(\beta)}$

Definición 1.3. Sea $(\Omega, \mathcal{S}, P)$ un espacio de probabilidad y α , $\beta \in \mathcal{S}$. α y β se dice que son **independientes** cuando: $P(\alpha \cap \beta) = P(\alpha)P(\beta)$

1.2.2. Teorema de Bayes

Como último paso antes de introducir las variables aleatorias, se va a presentar el teorema que da nombre a la teoría de las redes Bayesianas. Este teorema es consecuencia de las definiciones anteriormente mostradas de la probabilidad condicionada. El enunciado se basa en lo recogido por el libro *All Of Statistics* [Wasserman, 2013].

Teorema 1.1. *Sea $(\Omega, \mathcal{S}, P)$ un espacio de probabilidad y $A_1, ..., A_k$ una partición de Ω tal que $P(A_i) > 0$ para cada $i \in \{1, ..., k\}$. Entonces, para cualquier evento B tal que $P(B) > 0$; tenemos que, para cada $i \in \{1, ..., k\}$:*

$$P(A_i|B) = \frac{P(B|A_i)P(A_i)}{\sum_{j=1}^{k} P(B|A_j)P(A_j)}$$

Demostración. Por la definición de probabilidad condicionada se da la ecuación (1.1).

$$P(A_i|B) = \frac{A_i \cap B}{P(B)} = \frac{P(B|A_i)P(A_i)}{P(B)} \tag{1.1}$$

Como los conjuntos $A_1, ..., A_k$ son una partición de Ω, se tiene que:

$$B = \cup_{j=1}^{k}(A_j \cap B) \Rightarrow P(B) = \sum_{j=1}^{k} P(A_j \cup B) = \sum_{j=1}^{k} P(B|A_j)P(A_j) \tag{1.2}$$

Usando las ecuaciones (1.1) y (1.2) se demuestra el teorema.

$$P(A_i|B) = \frac{P(B|A_i)P(A_i)}{P(B)} = \frac{P(B|A_i)P(A_i)}{\sum_{j=1}^{k} P(B|A_j)P(A_j)}$$

Del teorema de Bayes, se deduce fácilmente el siguiente corolario para dos eventos cualesquiera.

Corolario 1.1. *Sea $(\Omega, \mathcal{S}, P)$ un espacio de probabilidad y A , B eventos cualesquiera tal que $P(B) > 0$, entonces:*

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

1.3. Variables aleatorias

En este momento del capítulo, ya se puede definir lo que es una variable aleatoria. Una variable aleatoria es una función que asigna un determinado valor al resultado de un experimento. Por ejemplo, en el experimento de tirar dos veces un dado, se puede definir una variable aleatoria cuyo posible valor sea $(1, 5)$ (primera tirada sale un 1 ,segunda tirada sale un 5). Si se formaliza convenientemente esta idea con la ayuda del libro *Probability Theory* [Borovkov, 2013], se obtiene la siguiente definición.

Sea $(\Omega, \mathcal{S}, P)$ un espacio de probabilidad y $\mathcal{B}$ espacio medible o de Borel.

Definición 1.4. Una **variable aleatoria** X es una función medible $X : (\Omega, \mathcal{S}) \longrightarrow (\mathbb{R}, \mathcal{B})$ en la que para todo $B \in \mathcal{B}$ se tiene que $X^{-1}(B) = \{w \in \Omega : X(w) \in B\} \in \mathcal{S}$.

Una vez definido el concepto de variable aleatoria, se define la función de distribución de una variable aleatoria como sigue.

Definición 1.5. Se llama **función de distribución** de la variable aleatoria X, a la función $F_X(x) = P(X < x)$.

La función de distribución cumple una serie de propiedades:

- $0 \leq F_X(x) \leq 1 \forall x \in \Omega$.
- $0 \leq F_X(x) \leq 1 \forall x \in \Omega$.
- $\lim_{x\to\infty} F_X(x) = 1$, $\lim_{x\to-\infty} F_X(x) = 0$
- $F_X(x)$ es no decreciente y continua por la derecha.

Generalizando más la definición de variable aleatoria se puede definir un vector de p variables aleatorias y su función de distribución.

Definición 1.6. Un **vector aleatorio** $X = (X_1, ..., X_p)$, es una aplicación medible $X : (\Omega, \mathcal{S}) \longrightarrow (\mathbb{R}^p, \mathcal{B}^p)$ en la que para todo $B \in \mathcal{B}^p$ se tiene que $X^{-1}(B) = \{w \in \Omega : X(w) \in B\} \in \mathcal{S}$.

Definición 1.7. Se define la **función de distribución** asociada al vector aleatorio $X = (X_1, ..., X_p)$, como la función $F_X(x) = P(X_1 < x_1, ..., X_n < x_n)$.

A continuación, se define la distribución conjunta entre una serie de eventos aleatorios. Intuitivamente, el significado de la distribución conjunta es la probabilidad de que esos eventos ocurran de forma simultánea.

Definición 1.8. Se define la **distribución conjunta** del vector aleatorio $X = (X_1, ..., X_p)$, como $P(X_1, ..., X_p) = P(X_1 = x_1, ..., X_n = x_n)$.

Ahora, se define un concepto muy importante en el estudio de las variables aleatorias; la esperanza matemática de una variable aleatoria. Esta definición es muy importante porque nos da información sobre el valor que se espera de una variable aleatoria.

Definición 1.9. La **esperanza** matemática de una variable aleatoria se define como el valor medio que toma dicha variable. Se denota como $\mathbb{E}$.

$$\mathbb{E}[X] = \sum_x x \cdot P[X = x]$$

Por último, se van a estudiar algunos conceptos básicos sobre las variables aleatorias; ya que, se usarán posteriormente en las redes Bayesianas.

1.3.1. Independencia

En este apartado, se va a exponer el significado de la independencia entre variables aleatorias. Uno de los conceptos claves en el desarrollo teórico de las Redes Bayesianas.

Definición 1.10. Sea $X = (X_1, ..., X_p)$ un vector aleatorio con función de distribución $F_X(x)$. Se dice que $X_1, ..., X_p$ son variables aleatorias **independientes** entre sí cuando se cumple que:

$$F_X(x) = F_{X_1}(x)...F_{X_p}(x)$$

Se denota la independencia entre las variables X_i y X_j como $X_i \perp X_j$.

1.3.2. Distribución Condicionada

La distribución de probabilidad de una variable condicionada a otra, surge como consecuencia directa de la Regla de Bayes. La definición se basa en información del libro *All Of Statistics* [Wasserman, 2013].

Definición 1.11. Sean X e Y variables aleatorias, definimos **la distribución de probabilidad de X condicionada a Y** como:

$$P(X = x|Y = y) = \frac{P(X = x, Y = y)}{P(Y = y)}$$

1.3.3. Independencia Condicional

Enlazando los dos conceptos anteriores, surge la independencia condicional. Concepto que abarca el conocimiento de independencia de una variable con otra, fijada una tercera.

Definición 1.12. Sean X, Y y Z variables aleatorias con distribución conjunta $P(X, Y, Z)$. Se dice que **X es condicionalmente independiente de Y dado Z** si :

$$P(X = x|Y = y) = P(X = x|Y = y, Z = z) \ \forall x, y, z : P(Z = z) > 0$$

Se denota la independencia condicional como $(X \perp Y|Z)$. Además, la independencia condicional tiene una serie de propiedades:

1. **Simetría**. $(X \perp Y|Z) \Rightarrow (Y \perp X|Z)$
2. **Descomposición**.$(X \perp Y, W|Z) \Rightarrow (X \perp Y|Z)$
3. **Unión débil**. $(X \perp Y, W|Z) \Rightarrow (X \perp Y|Z, W)$
4. **Contracción**. $(X \perp W|Z, Y)\&(X \perp Y|Z) \Rightarrow (X \perp Y, W|Z)$

1.3.4. Función de densidad

La función de densidad es importante pues proporciona el valor de probabilidad en un punto. Se define el concepto para vector aleatorio, sin pérdida de generalidad para una única variable; ya que, se trata de un caso particular de vector aleatorio de una única componente.

Definición 1.13. Se llamará **función de densidad** del vector aleatorio X, si existe, a la función f_X tal que:

$$F_X(x) = \int_{-\infty}^{x_p} ... \int_{-\infty}^{x_1} f_X(u_1, ..., u_p)\, du_1 ... du_p$$

Se dice entonces, que la distribución de X es continua.

Proposición 1.1. *Si f_X es una función continua, entonces se puede escribir como:*

$$f_X(x) = \frac{\partial^n}{\partial x_1 ... \partial x_p} F_X(x)$$

1.4. Distribuciones normales

La distribución normal juega un papel fundamental en las distribuciones continuas; como no podía ser de otra manera, en la representación de variables aleatorias continuas en redes Bayesianas también.

En el desarrollo de la teoría de redes Bayesianas se necesita conocer la distribución normal, y su extensión a múltiples variables. Las definiciones de la distribución normal se basan en el libro *Bayesian networks: with examples in R* [Scutari and Denis, 2014].

1.4.1. Distribución normal

La **distribución normal** de media μ y varianza σ de una variable, se denota por $N(\mu, \sigma^2)$. Una variable aleatoria X sigue una distribución normal si su función de densidad es de la siguiente forma.

$$f_X(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} exp\{-\frac{1}{2\sigma^2}(x-\mu)^2\}\ ,\ x, \mu \in \mathbb{R}, \sigma^2 > 0$$

Siendo $E[X] = \mu$ y $VAR[X] = \sigma^2$.

1.4.2. Distribución normal multivariante

La extensión multivariante de la distribución normal es llamada **distribución normal multivariante** o distribución gaussiana multivariante. Se denota por $N(\mu, \Sigma)$, donde μ es el vector de medias $k \times 1$ y Σ es la matriz de covarianzas semi-definida positiva de dimensión $k \times k$. Un vector aleatorio X sigue una distribución normal multivariante si su función de densidad es de la siguiente forma.

$$f_X(x; \mu, \Sigma^2) = \frac{1}{\sqrt{2\pi det(\Sigma)}} exp\{-\frac{1}{2}(x-\mu)^T \Sigma^{(}-1)(x-\mu))\ ,\ x, \mu \in \mathbb{R}^k$$

2. Teoría de Grafos

En este capítulo se van a definir conceptos básicos de teoría de grafos que se usarán en teoría de redes Bayesianas (RBs).

Lo expuesto en este capítulo se basa en conceptos descritos en el libro *Bayesian networks in R* [Nagarajan et al., 2013].

2.1. Grafos, Nodos y Arcos

Para comenzar, se va a definir el concepto de grafo; concepto base para definir la estructura de una red Bayesiana (RB).

Definición 2.1. Un **grafo** $\mathcal{G} = (V, A)$ consiste en un conjunto no vacío de nodos o vértices V y un conjunto finito de pares de vértices A llamados arcos, enlaces o lados.

Cada arco $a = (u, v)$ puede definirse como un par ordenado o desordenado de nodos, se dice que están conectados por el arco y que son adyacentes entre sí. Dado que son adyacentes, se dice que u y v también son vecinos. Si (u, v) es un par ordenado, se dice que u es la cola del arco y v la cabeza; entonces se dice que el arco está dirigido de u a v y se suele representar con una punta de flecha en v $(u \to v)$. Si (u, v) es desordenado, se dice simplemente que u y v son incidentes en el arco sin ninguna otra distinción. En este caso, se denominan comúnmente arcos o aristas no dirigidas.

Definición 2.2. Un grafo $\mathcal{G} = (V, A)$ se dice **dirigido** cuando sus arcos están ordenados y no dirigido cuando están desordenados.

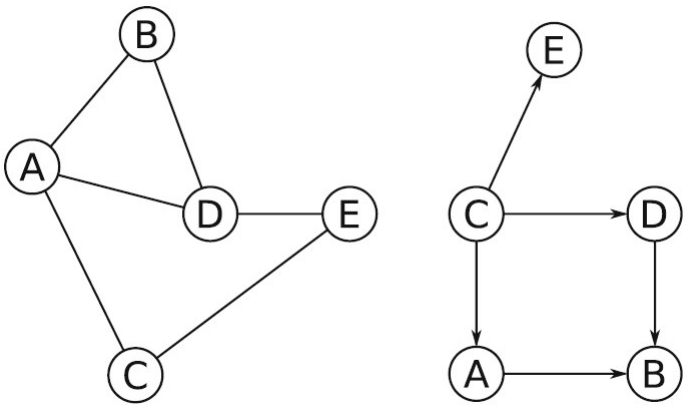

Figura 2.1.: Grafo no dirigido (izquierda), Grafo dirigido (derecha)

Si en un grafo dirigido se intercambian los arcos ordenados por arcos no ordenados se obtiene el **esqueleto** del grafo original dirigido.

2.2. La estructura de un grafo

Las relaciones de orden que se dan en un grafo dirigido, definen una estructura intrínseca al grafo. Sea un grafo dirigido $\mathcal{G} = (V, E)$, la estructura que conforma los arcos ordenados del conjunto E y sus nombres se ve representada en la figura 2.2 obtenida del libro *Bayesian networks in R*. [Nagarajan et al., 2013].

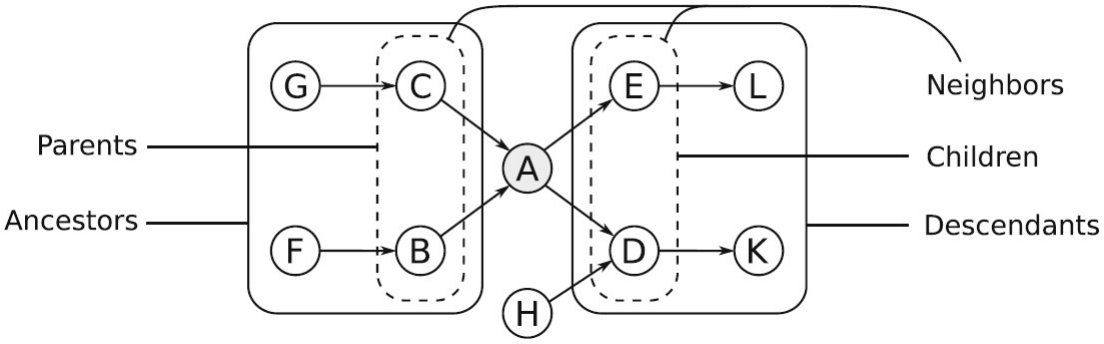

Figura 2.2.: Padres, ancestros, vecinos, hijos y descendientes del nodo A $\in V$

A continuación, se definen los conceptos de la figura anterior. Para ello, primero hay que definir un camino dirigido.

Definición 2.3. Sea un grafo $\mathcal{G} = (V, A)$. Un **camino dirigido** es una sucesión de aristas dirigidas con la misma orientación en la que cada cabeza de la arista debe coincidir con la cola de la arista siguiente. Por ejemplo, de G a K existe el camino dirigido $G \to C \to A \to D \to K$.

Definición 2.4. Sea un grafo $\mathcal{G} = (V, A)$ y $X \in V$. Los **padres** de X o Pa_X son todos los nodos $Y \in V$ tal que $(Y, X) \in A$ y es un par ordenado $(Y \to X)$.

Definición 2.5. Sea un grafo $\mathcal{G} = (V, A)$ y $X \in V$. Los **ancestros** de X son todos los nodos $Y \in V$ tal que existe un camino dirigido de Y a X.

Definición 2.6. Sea un grafo $\mathcal{G} = (V, A)$ y $X \in V$. Los **vecinos** de X son todos los nodos $Y \in V$ tal que existe un arco dirigido de Y a X o de X a Y.

Definición 2.7. Sea un grafo $\mathcal{G} = (V, A)$ y $X \in V$. Los **hijos** de X son todos los nodos $Y \in V$ tal que existe un arco dirigido de X a Y $(X \to Y)$.

Definición 2.8. Sea un grafo $\mathcal{G} = (V, A)$ y $X \in V$. Los **descendientes** de X son todos los nodos $Y \in V$ tal que existe un camino dirigido de X a Y.

Además, para la definición de red Bayesiana, se necesita definir a los no descendientes de un nodo X o $NonDescendants_X$, como aquellas variables que no son descendientes de X.

Definición 2.9. Un **grafo dirigido acíclico** (DAG) es un grafo dirigido que no tiene ciclos.

El grafo de la figura 2.2 es un grafo dirigido acíclico. Estos grafos nos proporcionan un orden local entre los nodos, orden que definiremos a continuación.

Definición 2.10. Sea un grafo $\mathcal{G} = (V, A)$. Un orden de los nodos $X_1, ..., X_n$ es un **orden topológico** si para todo arco $(X_i, X_j) \in A$ se tiene que $i < j$.

Tras la definición de DAG, ya se está preparado para estudiar la representación de una RB; pues se han expuesto los conceptos básicos necesarios tanto de probabilidad como de grafos.

3. Representación de la Red Bayesiana

> En este capítulo veremos las definiciones principales de la teoría de redes Bayesianas así como propiedades de la independencia de grafos y una introducción a mapas.

Me baso en el libro *Probabilistic Graphical Models* [Koller and Friedman, 2009] para definir conceptos de interés.

3.1. Redes Bayesianas

El objetivo para el que se crean las redes Bayesianas (RBs) es poder representar una distribución P sobre un **conjunto de variables aleatorias** $\boldsymbol{X = X_1, ..., X_n}$. Incluso en el caso más simple donde estas variables son binarias, una distribución conjunta requiere la especificación de $2^n - 1$ números (las probabilidades de las 2^n asignaciones diferentes de los valores $x_1, ..., x_n$). Para casi todo n (menos un n muy pequeño), la representación explícita de la distribución conjunta es inimaginable desde varios puntos.

Desde el punto de vista computacional, es muy costoso de manipular y, en general, demasiado grande para almacenarlo en la memoria. Desde el punto de vista cognitivo, es imposible adquirir tantos números de un experto humano; además, los números son muy pequeños y no corresponden a eventos que las personas puedan contemplar razonablemente.

Desde el punto de vista estadístico, si se quiere aprender la distribución a partir de los datos, se necesitan cantidades inmensas de datos para estimar tantos parámetros de forma sólida. Estos problemas fueron el principal obstáculo para la adopción de métodos probabilísticos para sistemas expertos hasta el desarrollo de las RBs.

En este punto, después de haber introducido los conceptos esenciales de teoría de grafos y teoría de la probabilidad, se puede proceder a definir la estructura de una RB.

Definición 3.1. Sea $\mathcal{G}$ un DAG. Denotaremos como $Pa^{\mathcal{G}}_{X_i}$ a los padres del nodo X_i en el grafo $\mathcal{G}$.

Definición 3.2. Sea $\mathcal{G}$ un DAG. Denotaremos como $NonDescendants_{X_i}$ a las variables en el grafo $\mathcal{G}$ que no sean descendientes de X_i.

Definición 3.3. Una **estructura de red Bayesiana** $\mathcal{G}$ es un DAG cuyos nodos representan variables aleatorias $X_1, ..., X_n$. Entonces $\mathcal{G}$ codifica el conjunto de relaciones de independencia condicionales, llamadas independencias locales, y denotadas como $\mathcal{I}_l(\mathcal{G})$:

$$\forall X_i : (X_i \perp NonDescendants_{X_i} | Pa^{\mathcal{G}}_{X_i}).$$

En otras palabras, cada nodo X_i es condicionalmente independiente de sus no descendientes dados sus padres.

3.1.1. Grafos y distribuciones

Como ya se ha comentado, el objetivo de las RBs es representar una distribución P sobre un conjunto de variables aleatorias. En este apartado, se demuestra que una distribución P satisface las independencias locales asociadas con un grafo $\mathcal{G}$, si y solo si P es representable como un conjunto de distribuciones de probabilidad condicionada (DPCs) asociadas con el grafo $\mathcal{G}$. Para ello, se comienza definiendo algunos conceptos nuevos.

Definición 3.4. Sea P una distribución sobre X. Se define $\boldsymbol{\mathcal{I}(P)}$ como el conjunto de relaciones de independencias condicionales de la forma $(X \perp Y|Z)$ que pertenecen a P.

Con este nuevo concepto, se puede reescribir la afirmación de que P satisface las independencias locales asociadas con el grafo $\mathcal{G}$, simplemente escribiendo que $\mathcal{I}_l(\mathcal{G}) \subseteq \mathcal{I}(P)$.

Definición 3.5. Sea $\mathcal{K}$ un grafo con conjunto de independencias $\mathcal{I}_l(\mathcal{K})$. Se dice que $\mathcal{K}$ es un **I-mapa** para un conjunto de independencias $\mathcal{I}$ si $\mathcal{I}_l(\mathcal{K}) \subseteq \mathcal{I}$. Además, si $\mathcal{I} = \mathcal{I}(P)$, para alguna distribución P, se dice que $\mathcal{K}$ es un I-mapa para P

Entonces, usando las dos definiciones anteriores, se obtiene que $\mathcal{G}$ es un I-mapa para P; ya que, $\mathcal{I}_l(\mathcal{G}) \subseteq \mathcal{I}(P)$. A continuación, se va a usar el concepto de $Pa_{X_i}^{\mathcal{G}}$ para definir la distribución conjunta en una RB.

Definición 3.6. Sea $\mathcal{G}$ una estructura de RB sobre las variables $X_1, ..., X_n$. Se dice que una distribución P sobre el mismo espacio **factoriza según** $\mathcal{G}$ si puede ser expresado como el siguiente producto:

$$P(X_1, ..., X_n) = \prod_{i=1}^{n} P(X_i|Pa_{X_i}^{\mathcal{G}})$$

Esta ecuación es llamada la regla de la cadena para RBs. El factor $P(X_i|Pa_{X_i}^{\mathcal{G}})$ es denominado distribución condicional de probabilidad (DPCs) o modelo probabilístico local.

Ahora, después de la definición de todos estos conceptos; se puede definir formalmente el concepto de RB.

Definición 3.7. Se denomina **Red Bayesiana (RB)** al par $\mathcal{B} = (\mathcal{G}, P)$ donde P factoriza sobre $\mathcal{G}$ y donde P es especificada como un conjunto de DPCs asociadas con los nodos de $\mathcal{G}$. La distribución P se denota como $P_{\mathcal{B}}$.

Finalmente, con los dos teoremas siguientes, se demuestra que una distribución P satisface las independencias locales asociadas con un grafo $\mathcal{G}$, si y solo si P es representable como un conjunto de distribuciones condicionales de probabilidad (DPCs) asociadas con el grafo $\mathcal{G}$.

Teorema 3.1. *Sea $\mathcal{G}$ una estructura de RB sobre un conjunto de variables aleatorias X. Y sea P la distribución conjunta sobre el mismo espacio. Si $\mathcal{G}$ es un I-mapa de P, entonces P factoriza sobre $\mathcal{G}$.*

Demostración. Se puede asumir sin pérdida de generalidad que las variables $X_1, ..., X_n$ del conjunto X tienen un orden topológico (2.10). Por tanto, usando la regla de la cadena para RBs se obtiene lo siguiente:

$$P(X_1, ..., X_n) = \prod_{i=1}^{n} P(X_i|X_1, ..., X_{i-1})$$

Considerando uno de los factores $P(X_i|X_1,...,X_{i-1})$. Como $\mathcal{G}$ es un I-mapa de P, se tiene que $(X_i \perp NonDescendants_{X_i}|Pa^{\mathcal{G}}_{X_i}) \in \mathcal{I}(P)$. Asumiendo que todos los padres de X_i están en el conjunto $X_1,...,X_{i-1}$. Entonces, ninguno de los descendientes de X_i están en ese conjunto. Por lo que:

$$\{X_1,...,X_{i-1}\} = Pa^{\mathcal{G}}_{X_i} \cup Z \;\; donde\; Z \subseteq NonDescendants_{X_i}$$

A partir de las independencias locales de X_i y de la propiedad de la descomposición , se tiene que $(X_i \perp Z|Pa^{\mathcal{G}}_{X_i})$. Entonces:

$$P(X_i|X_1,...,X_{i-1}) = P(X_i|Pa^{\mathcal{G}}_{X_i})$$

Aplicando esta transformación para todos los factores de la regla de la cadena para RBs obtenemos que P factoriza sobre $\mathcal{G}$.

Teorema 3.2. *Sea $\mathcal{G}$ una estructura de red Bayesiana sobre un conjunto de variables aleatorias X. Y sea P la distribución conjunta sobre el mismo espacio. Si P factoriza sobre $\mathcal{G}$, entonces $\mathcal{G}$ es un I-mapa de P.*

3.1.2. Ejemplo red asia

Para clarificar lo expuesto, considero la red Bayesiana conocida como asia, red que se describió por primera vez en el trabajo de *Local computations with probabilities on grafical structures and their aplication to expert systems* [Lauritzen and Spiegelhalter, 1988] y cuya representación se presenta en la figura siguiente.

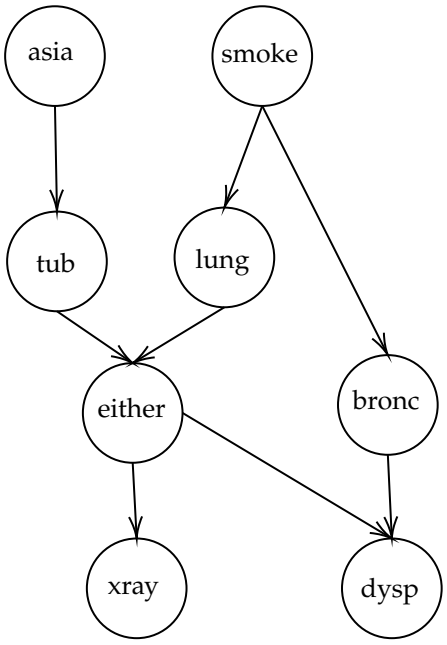

Figura 3.1.: Red Asia

La red asia representa el conocimiento sobre la disnea (dysp). La disnea es la falta de aire o dificultad para respirar. Las posibles causas que se estudian en la red son la tuberculosis

(tub), el cáncer de pulmón (lung) o la bronquitis (bronc). La probabilidad de tener tuberculosis se puede ver incrementada mediante los viajes a Asia (asia). Por otra parte, ser fumador (smoke) incrementa las probabilidades de sufrir cáncer de pulmón o bronquitis. La influencia de las variables tub y lung se combinan en la variable either porque ambas se pueden detectar mediante una prueba de rayos (xray). Finalmente, las variables either y bronc son las que influyen directamente sobre dysp. Todas las variables son de dominio binario (tiene tuberculosis si/no, tiene cáncer de pulmón si/no,...).

Una vez definido el grafo, se pueden observar las variables y las relaciones que existen en el problema. Finalmente, queda por definir los valores que toman estas relaciones especificando las distribuciones de probabilidad de las mismas.

- $P(asia)$. Probabilidad de haber viajado a Asia.
- $P(smoke)$. Probabilidad de fumar
- $P(tub|asia)$. Probabilidad de tener tuberculosis, habiendo viajado a Asia.
- $P(lung|smoke)$. Probabilidad de tener cáncer de pulmón siendo fumador.
- $P(either|tub, lung)$. Probabilidad que combina la información de tener tuberculosis y cáncer de pulmón.
- $P(bronc|smoke)$. Probabilidad de tener bronquitis siendo fumador.
- $P(xray|either)$. Probabilidad de prueba de rayos x sabiendo la variable either.
- $P(dysp|either, bronc)$. Probabilidad de disnea condicionada a las variables either y bronc.

3.2. Independencia en grafos

Las propiedades de dependencia e independencia de una distribución son cruciales para entender su comportamiento. Como se ha expuesto en la sección anterior, una estructura de grafo $\mathcal{G}$ codifica un conjunto de independencias condicionales $\mathcal{I}_l(\mathcal{G})$. Sabiendo únicamente que una distribución P factoriza sobre $\mathcal{G}$, podemos concluir que satisface $\mathcal{I}_l(\mathcal{G})$. Entonces, cabría preguntarse la existencia de otras independencias que se puedan obtener directamente de $\mathcal{G}$. Esta cuestión es la que vamos a resolver en este apartado.

3.2.1. D-separación

El objetivo es ver cuando se puede garantizar que las independencias condicionales $(X \perp Y|Z)$ se mantienen en una distribución asociada a una RB $\mathcal{G}$. Por tanto, se tiene que estudiar cuando es posible que X pueda influir en Y dado Z. Aparecen cuatro posibles relaciones representadas en la figura 3.2.

Definición 3.8. Se le llama **v-estructura**, a la estructura de la forma $X \rightarrow Z \leftarrow Y$. Figura 3.2 (d).

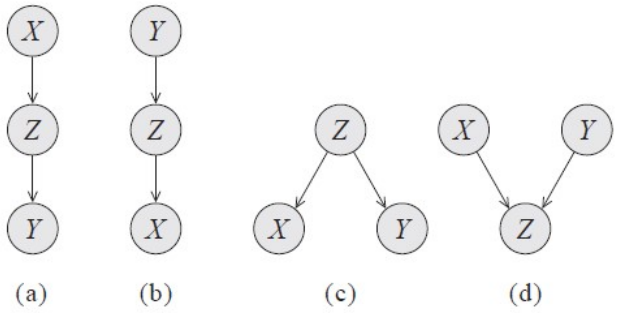

Figura 3.2.: (a) efecto causal indirecto, (b) efecto evidencial indirecto, (c) Causa común, (d) Efecto común

Definición 3.9. Sea $\mathcal{G}$ una estructura de red Bayesiana y $X_1 \rightleftharpoons ... \rightleftharpoons X_n$ un camino en $\mathcal{G}$. Sea Z un conjunto de variables observadas. El camino $X_1 \rightleftharpoons ... \rightleftharpoons X_n$ es activo dado Z si:

- Para cualquier v-estructura $X_{i-1} \rightarrow X_i \leftarrow X_{i+1}$, entonces X_i o uno de sus descendientes pertenece a Z.
- Ningún otro nodo del camino está en Z.

Observación 3.1. Si X_1 o X_n pertenecen a Z, el camino no es activo.

Definición 3.10. Sea $\mathcal{G}$ una estructura de red Bayesiana y sea X,Y,Z tres conjuntos de nodos en $\mathcal{G}$. Se dice que X e Y están **d-separados** dado Z, y se denota como $d - sep_{\mathcal{G}}(X;Y|Z)$, si no hay un camino activo entre cualquier nodo $X_i \in X$ y $Y_i \in Y$ dado Z. Se denota por $\mathcal{I}(\mathcal{G})$ al conjunto de independencias que corresponden con la d-separación:

$$\mathcal{I}(\mathcal{G}) = \{(X \perp Y|Z) : d - sep_{\mathcal{G}}(X;Y|Z)\}$$

Este conjunto es llamado también como las **independencias globales de Markov**.

3.2.2. Robustez y completitud

Una de las propiedades que se induce de la d-separación es la **robustez**. Ante dos nodos X e Y que están d-separados dado un nodo Z, se puede garantizar que son condicionalmente independientes dado Z.

Teorema 3.3. *Si una distribución P factoriza de acuerdo con $\mathcal{G}$, entonces $\mathcal{I}(\mathcal{G}) \subseteq \mathcal{I}(P)$.*

Otra propiedad es la **completitud**. Para exponer esta propiedad se necesita definir en primer lugar el concepto de fidelidad.

Definición 3.11. Una distribución P es **fiel** a $\mathcal{G}$, si para cualquier $(X \perp Y|Z) \in \mathcal{I}(P)$, entonces $d - sep_{\mathcal{G}}(X;Y|Z)$. En otras palabras, cualquier independencia en P es reflejada en las propiedades de la d-separación del grafo.

En este momento, ya se puede exponer porque se induce la **completitud** del concepto de d-separación. Para cualquier distribución P que factoriza sobre $\mathcal{G}$, se tiene que P es fiel a $\mathcal{G}$; es decir, si X e Y son no d-separados dado Z en $\mathcal{G}$, entonces X e Y son dependientes en todas las distribuciones P que factorizan sobre $\mathcal{G}$.

Teorema 3.4. *Sea $\mathcal{G}$ una estructura de red Bayesiana. Si X e Y son no d-separados dado Z en $\mathcal{G}$, entonces X e Y son dependientes dado Z en alguna distribución de P que factorice sobre $\mathcal{G}$.*

Demostración. La prueba construye una distribución P que hace que X e Y estén correlados. Como X e Y son no d-separados, existe un camino activo $U_1, ..., U_k$ entre ellos. Definimos DPCs para las variables del camino; para que cada pareja U_i, U_{i+1} sean correladas. En el caso de una v-estructura $U_i \rightarrow U_{i+1} \leftarrow U_{i+2}$, definimos la DPC de U_{i+1} de manera que se garantice la correlación, y también se definan los DPCs del camino hacia algún nodo descendente; de forma que podamos garantizar que se active la correlación entre U_i y U_{i+2}. Todos los demás DPCs del gráfo se eligen para que sean uniformes, y así la construcción garantiza que la influencia sólo fluye a lo largo de este único camino, evitando los casos en los que la influencia de dos (o más) caminos se cancela.

Teorema 3.5. *Para casi todas las distribuciones P que factorizan sobre $\mathcal{G}$ (para todas las distribuciones excepto un conjunto de medida cero en el espacio de las parametrizaciones DPC), tenemos que $\mathcal{I}(\mathcal{G}) = \mathcal{I}(P)$.*

Demostración. Cada independencia condicional es un conjunto de igualdades polinómicas sobre el espacio de las parametrizaciones DPC. Una propiedad básica de los polinomios es que un polinomio es idénticamente cero o su conjunto de raíces tiene medida nula. El teorema 3.4 implica que los polinomios correspondientes a las relaciones fuera de $\mathcal{I}(\mathcal{G})$ no pueden ser idénticamente cero. Por lo tanto, el conjunto de distribuciones P, que exhiben cualquiera de estas relaciones de independencia tiene medida cero. El conjunto de distribuciones que no satisfacen $\mathcal{I}(\mathcal{G}) = \mathcal{I}(P)$ es la unión de los conjuntos con medida cero. La unión de un número finito de conjuntos con medida cero es un conjunto de medida cero, probando el resultado.

Con este resultado se afirma que para casi todas las parametrizaciones P del grafo G ;es decir, para casi todas las elecciones posibles de DPCs para las variables; la d-separación caracteriza con precisión las independencias que se mantienen para P.

En el libro de *Probabilistic Graphical Models* [Koller and Friedman, 2009] se encuentra un **algoritmo** para determinar la condición de d-separación.

3.2.3. I-equivalencia

La noción de $\mathcal{I}(\mathcal{G})$ especifica un conjunto de independencias condicionales que se asocian a un grafo. Con esto se puede abstraer los detalles de la estructura del grafo, considerándolo puramente como un conjunto de relaciones de independencia. Una importante conclusión que aporta esta perspectiva es que dos estructuras de RBs pueden ser equivalentes si codifican las mismas independencias.

Definición 3.12. Dadas dos estructuras de grafos $\mathcal{K}_1$ y $\mathcal{K}_2$ sobre X , las dos estructuras son **I-equivalentes** si $\mathcal{I}(\mathcal{K}_1) = \mathcal{I}(\mathcal{K}_2)$.

El conjunto de todos los grafos sobre X es particionado en un conjunto de clases de I-equivalencias mutuamente exhaustivas y excluyentes, conjunto inducido mediante la relación de I-equivalencia.

La I-equivalencia nos dice que una distribución P que pueda ser factorizada por un grafo, también lo será con el otro. Sin embargo, no nos da información sobre la dirección de las dependencias. Es decir, podemos saber que en la distribución $P(X, Y)$, X e Y están correlados,

sin embargo, no sabemos nada sobre si la dependencia es $X \rightarrow Y$ o $X \leftarrow Y$. Para caracterizar la I-equivalencia, se necesita conocer el concepto de esqueleto para una estructura de RB.

Definición 3.13. El **esqueleto de una estructura de RB** $\mathcal{G}$ sobre X es un grafo no dirigido sobre X que contiene un lado no dirigido $\{X, Y\}$ por cada lado dirigido (X, Y) de $\mathcal{G}$. En la figura 3.2, los cuatro grafos tienen el mismo esqueleto.

Teorema 3.6. *Sean $\mathcal{G}_1$ y $\mathcal{G}_2$ dos grafos sobre X. Si dos grafos $\mathcal{G}_1$ y $\mathcal{G}_2$ tienen el mismo esqueleto y las mismas v-estructuras, ambos son I-equivalentes.*

Cabe resaltar que el recíproco no es cierto, hay grafos I-equivalentes con distintos conjuntos de v-estructuras. Para poder tener equivalencia hay que usar una hipótesis más estricta a la v-estructura.

Definición 3.14. Una v-estructura $X \rightarrow Z \leftarrow Y$ es una **inmoralidad**, si no existe un lado entre X e Y.

Teorema 3.7. *Sean $\mathcal{G}_1$ y $\mathcal{G}_2$ dos grafos sobre X. Dos grafos $\mathcal{G}_1$ y $\mathcal{G}_2$ tienen el mismo esqueleto y las mismas inmoralidades si y sólo si son I-equivalentes.*

3.3. Mapas

Se ha demostrado, que si P se factoriza sobre $\mathcal{G}$, se pueden derivar un gran conjunto de relaciones de independencia que se mantienen para P, simplemente examinando $\mathcal{G}$. Este resultado conduce inmediatamente a la idea de que se puede utilizar un grafo como una forma de revelar la estructura en una distribución. En particular, se pueden probar las independencias en P construyendo un grafo $\mathcal{G}$ que represente a P y probando la d-separación en $\mathcal{G}$. Tener este grafo $\mathcal{G}$ tiene otras consecuencias importantes, en términos de reducir el número de parámetros requeridos para aprender la distribución y disminuir la complejidad de realizar inferencia en el grafo.

3.3.1. I-mapa minimal

Una primera aproximación para encontrar el grafo $\mathcal{G}$ es escoger un I-mapa de la distribución P. Está claro que el grafo de todos los lados es un grafo posible; por tanto, habría que eliminar lados redundantes, para eso se define la idea de I-mapa minimal.

Definición 3.15. Un grafo $\mathcal{K}$ es un **I-mapa minimal** para un conjunto de independencias $\mathcal{I}$, si es un I-mapa y si al eliminar algún lado de $\mathcal{K}$ ya no es un I-mapa para $\mathcal{I}$.

En el libro de *Koller y Nir Friedman*[Koller and Friedman, 2009] encontramos un algoritmo para determinar la condición de I-mapa minimal.

3.3.2. I-mapa perfecto

Los I-mapas minimales no capturan necesariamente todas las independencias que se dan en la distribución. Por esta razón aparece la idea de I-mapa perfecto (P-mapa).

Definición 3.16. Un grafo $\mathcal{K}$ es un **P-mapa** para un conjunto de independencias $\mathcal{I}$ si $\mathcal{I}(\mathcal{K}) = \mathcal{I}$.

Definición 3.17. Un grafo $\mathcal{K}$ es un **P-mapa** para un una distribución P si $\mathcal{I}(\mathcal{K}) = \mathcal{I}(P)$.

Por desgracia, nos encontramos ante el inconveniente de que no todas las distribuciones tienen un P-mapa.

En el libro de *Koller y Nir Friedman*[Koller and Friedman, 2009] encontramos un algoritmo para encontrar P-mapas bajo ciertas hipótesis.

4. Modelos Probabilísticos Locales

> En este capítulo se va a profundizar en el estudio de las distribuciones de probabilidad condicionada (DPCs). Explicando las diferentes representaciones de las DPCs mediante los modelos probabilísticos locales.

Los conceptos que se describen en este capitulo se basan en el libro *Probabilistic Graphical Models* [Koller and Friedman, 2009]

4.1. Introducción

En los capítulos anteriores se ha estudiado la representación de las propiedades globales de independencia. Estas propiedades de independencia permiten factorizar una distribución conjunta de alta dimensión en un producto de DPCs, factores de menor dimensión. Hasta ahora, se ha ignorado la representación de estos factores.

En este capítulo se estudiarán principalmente formas de representar espacios con variables discretas, pero haré referencia a las variables continuas en una sección final.

4.2. DPCs en formato tabla

En espacios compuestos por variables aleatorias discretas siempre podemos representar las DPCs en formato de tabla. En este formato, la DPC $P(X|Pa_X)$ se codifica como una tabla que contiene una entrada para cada asignación de X y Pa_X. Para que la tabla esté bien definida se requiere que todos los valores sean no negativos y que además, para cada Pa_X se dé:

$$\sum_{x \in Val(X)} P(x|Pa_X) = 1$$

Sin embargo, la representación en formato de tablas tiene un inconveniente, el número de las entradas de la tabla crece exponencialmente con el número de padres. La dimensión de la tabla depende tanto de los valores que puede tomar X como los valores que pueden tomar los padres de X. Por tanto, la dimensión se puede calcular como $|Val(X)||Val(Pa_X)|$.

A continuación, se muestra un ejemplo simple de DPC en formato tabla para dos variables binarias X e Y con relación de independencia $X \to Y$.

Y	x^0	x^1
y^0	$P(y^0\|x^0)$	$P(y^0\|x^1)$
y^1	$P(y^1\|x^0)$	$P(y^1\|x^1)$

Tabla 4.1.: Ejemplo DPC en formato tabla

4.3. DPCs deterministas

Otra alternativa al formato de tablas por su sencillez es la representación de DPCs deterministas. Una DPC determinista se puede representar cuando una variable X es una función determinista de sus padres, o lo que es lo mismo; existe una función $f : Val(Pa_X) \longrightarrow Val(x)$ tal que:

$$P(x|Pa_X) = \begin{cases} 1 & si \quad x = f(pa_X) \\ \\ 0 & \text{en otro caso} \end{cases}$$

De esta forma, la DPC solo toma los valores 1 o 0. Aunque no todas se pueden escribir como funciones deterministas de sus padres.

4.4. DPCs representados mediante árboles

Una forma muy natural de representar DPCs es mediante un árbol, donde las hojas son las posibles distribuciones y el camino hasta la hoja aporta el contexto.

Definición 4.1. Un **árbol-DPC** que representa una DPC para la variable X es un grafo en forma de árbol, en el que cada nodo del árbol es un nodo hoja o un nodo interior. Cada hoja está etiquetada con una distribución $P(X)$. Cada nodo interior está etiquetado con una variable $Z \in Pa_X$ y tiene un conjunto de arcos salientes a sus hijos, cada uno asociado con un único valor asignado a la variable $Z = z_i$ para $z_i \in Val(Z)$.

Definición 4.2. Una rama de un árbol-DPC es un camino desde la raíz a un nodo hoja. Sin que la rama tenga dos nodos interiores asociados a la misma variable.

Una rama del árbol

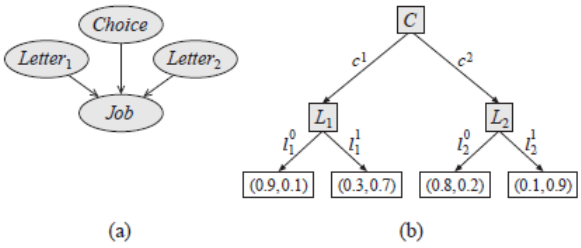

Figura 4.1.: (a) Red de ejemplo. (b) Árbol-DPC para $P(J|C, L_1, L_2)$.

En la figura 4.1 se muestra un ejemplo de RB para conocer la probabilidad de conseguir un trabajo y la DPC $P(J|C, L_1, L_2)$ representada mediante un árbol-DPC. Asumiendo que todas las variables son binarias, se puede, por ejemplo, obtener la probabilidad de que $P(J = j^0|C = c^1, L_1 =_1^0) = 0.9$.

4.5. DPCs definidas mediante reglas

Como se ha visto, los árboles capturan la DPC en una estructura de datos de forma global. En muchos casos, es más fácil razonar utilizando una DPC si se descompone la estructura de dependencias en elementos más pequeños.

Definición 4.3. Una regla ρ es un par $\langle c; p \rangle$ donde c es una asignación a algún subconjunto de variables C y $p \in [0, 1]$. Decimos que C es el marco de ρ, y lo denotamos como $Scope[\rho]$

Esta nueva notación en forma de regla, se basa en el concepto de rama de un árbol-DPC. Gracias a las reglas, se consigue descomponer un árbol-DPC en sus elementos básicos.

Definición 4.4. Un DPC $P(X|Pa_X)$ basado en reglas es un conjunto de reglas $\mathcal{R}$ tal que:

- Para cada $\rho \in \mathcal{R}$, se tiene que $Scope[\rho] \subseteq \{X\} \cup Pa_X$.
- Para cada asignación (x, u) de $\{X\} \cup Pa_X$, se tiene una regla $\langle c; p \rangle \in \mathcal{R}$ tal que c es compatible con (x, u). En este caso decimos que $P(X = x|Pa_X = u) = p$.
- La DPC resultante $P(X|U)$ es una DPC bien definida en la que:

$$\sum_x P(x|u) = 1$$

4.6. Influencia causal

En este apartado describo un nuevo tipo de modelo probabilístico local en el que se considera una variable Y cuya distribución depende de un conjunto de variables $X_1, .., X_k$. En general, las variables X_i pueden interactuar entre ellas de formas complejas. Sin embargo, en muchos casos, la influencia combinada de las X_i sobre Y es una simple combinación de la influencia de cada una de las variables X_i sobre Y de forma aislada. En otras palabras, cada una de las X_i influye en Y de forma independiente, y la influencia de varias de ellas simplemente se combina de alguna manera. Se estudiarán dos tipos de modelos, el modelo noisy-or y los modelos lineales generalizados.

4.6.1. Modelo Noisy-OR

Para este modelo se necesita que las variables aleatorias sean binarias. A continuación defino formalmente el modelo.

Definición 4.5. Sea Y una variable aleatoria binaria con k padres $X_1, ..., X_k$ que toman valores binarios. La DPC $P(Y|X_1, ...X_k)$ se representa mediante el **modelo Noisy-OR** si hay $k + 1$ parámetros $\lambda_0, ..., \lambda_k$ tal que:

$$P(y^0|X_1, ..., X_k) = (1 - \lambda_0) \prod_{i:X_i = x_i^1} (1 - \lambda_i)$$

$$P(y^1|X_1, ..., X_k) = 1 - P(y^0|X_1, ..., X_k) = 1 - [(1 - \lambda_0) \prod_{i:X_i = x_i^1} (1 - \lambda_i)]$$

En esta expresión, se entiende que i es el valor del subíndice de x cuando este toma el segundo valor de los dos posibles. Podemos reinterpretar la fórmula si notamos x_i^1 como 1 y x_i^0 como 0 de la forma siguiente.

$$P(y^0|X_1,...,X_k) = (1-\lambda_0)\prod_{i=1}^{k}(1-\lambda_i)^{x_i}$$

El modelo noisy-or proporciona una buena aproximación para problemas en los que la variable a estudiar tiene una gran cantidad de padres y su domino es binario.

4.6.2. Modelos lineales generalizados

En esta sección se van a exponer modelos que definen distribuciones de probabilidad $P(Y|X_1,...,X_k)$ donde Y toma valores en un espacio discreto finito. Este caso se divide en dos, cuando las variables son binarias (Binomial) o cuando las variables toman más valores (extendiendo el caso Binomial al caso Multinomial).

4.6.2.1. Variables binarias

En este caso se expone la DPC de Y dadas $X_1,...,X_k$ variables aleatorias como el efecto que produce las variables X_i sobre Y. Este efecto se puede resumir en la función lineal construida mediante la combinación lineal de los X_i y unos pesos w_i asociados a cada variable. Interpretando como en el modelo anterior x_i^1 como 1 y x_i^0 como 0.

$$f(X_1,...,X_k) = \sum_{i=1}^{k} w_i X_i$$

Hay que abordar la cuestión de cuando se transiciona de un valor de Y a otro. Para esto se trabaja con un umbral fijo; es decir, cuando $f(X_1,...,X_k) \geqslant w_0$ en Y se produce un cambio. Esta idea es fácil de implementar en la función anterior de la forma siguiente.

$$f(X_1,...,X_k) = w_0 + \sum_{i=1}^{k} w_i X_i$$

Además, para proporcionar una transición suave del valor de Y, una opción muy común es usar la función sigmoide:

$$sigmoid(z) = \frac{e^z}{1+e^z}$$

Esta idea se formaliza de la siguiente forma:

Definición 4.6. Sea Y una variable aleatoria binaria con k padres $X_1,...,X_k$ que toman valores numéricos $\{0,1\}$. La DPC $P(Y|X_1,...,X_k)$ es una **DPC logística** si hay $k+1$ pesos $w_0,...,w_k$ tal que:

$$P(y^1|X_1,...,X_k) = sigmoid(w_0 + \sum_{i=1}^{k} w_i X_i)$$

4.6.2.2. Variables multivaluadas

Finalmente, se extiende la DPC logística al caso multivaluado donde Y toma múltiples valores $y^1,...,y^m$. El concepto se extiende de la siguiente forma.

Definición 4.7. Sea Y una variable aleatoria que toma m valores con k padres $X_1, ..., X_k$ que toman valores numéricos. La DPC $P(Y|X_1, ..., X_k)$ es una **DPC logística multinomial** si para cada $j = 1, ..., m$, hay k+1 pesos $w_{j,0}, ..., w_{j,k}$ tal que:

$$l_j(X_1, ..., X_k) = w_{j,0} + \sum_{i=1}^{k} w_{j,i} X_i$$

$$P(y^j|X_1, ..., X_k) = \frac{exp(l_j(X_1, ..., X_k))}{\sum_{j'=1}^{m} exp(l_{j'}(X_1, ..., X_k))}$$

4.7. Variables continuas

Todo el capítulo se ha trabajado con variables discretas, pero las variables aleatorias también se pueden modelar mediante espacios continuos. Una supuesta solución podría ser discretizar las variables continuas para aplicar lo anteriormente expuesto; pero con esto, se pierde mucha información relevante al problema en cuestión. Por lo que, aparecen otras técnicas para el tratado de variables continuas.

Para definir la DPC en variables continuas se necesita las distribuciones normales o gaussianas que aparecen en el capítulo 1.4.

Definición 4.8. Sea Y una variable continua con padres continuos $X_1, ..., X_k$. Decimos que Y está representado mediante un modelo gaussiano lineal si existen parámetros $\beta_0, ..., \beta_k$ y σ^2 tal que:

$$P(Y|x_1, ..., x_k) = N(\beta_0 + \beta_1 x_1 + ... + \beta_k x_k; \sigma^2)$$

Desde otro punto de vista, esta definición dice que Y es una función lineal. Es decir; $Y = \beta_0 + \beta_1 x_1 + ... + \beta_k x_k + \epsilon$ donde ϵ es la variable aleatoria gaussiana de media 0 y varianza σ^2 que modela el ruido del sistema.

4.8. Modelos híbridos

Por último, pueden existir situaciones en las que sean necesarias variables discretas y variables continuas, pero con lo ya estudiado es fácil dar una solución a este problema. Nos encontramos ante dos posibilidades: variables continuas con padres discretos o/y continuos; o variables discretas con padres discretos o/y continuos.

En el caso de que la variable a estudiar sea discreta, se usaría el modelo DPC logístico visto anteriormente. Sin embargo, si la variable es continua, lo que se hace es generar una gaussiana lineal para cada combinación de valores de los padres discretos.

5. Inferencia

> En este capítulo se va a tratar el problema de inferencia sobre los modelos gráficos probabilísticos y en concreto en una red Bayesiana.

Una vez se ha profundizado la representación de CPDs, se puede abordar el problema de inferencia sobre RBs. En el capítulo se definen conceptos basados en el libro *Probabilistic Graphical Models* [Koller and Friedman, 2009].

5.1. Introducción

En este capítulo se va a mostrar el problema de inferencia en los modelos gráficos. Tanto la estructura del grafo como las relaciones de independencia van a permitir hacer inferencia de forma manejable.

Para ello, hay que centrarse en la consulta más común: la consulta de probabilidad condicionada. Por la definición de probabilidad condicionada, se conoce:

$$P(Y|E=e) = \frac{P(Y,e)}{P(e)}$$

Cada una de las instancias del numerador es una expresión de probabilidad $P(y,e)$, que puede calcularse sumando todas las entradas de la distribución conjunta que corresponden a asignaciones consistentes con y, e. Más concretamente, $W = \mathcal{X} - Y - E$ son las variables aleatorias que no son ni consulta ni evidencia. Entonces:

$$P(y,e) = \sum_{w} P(y,e,w)$$

Como W, Y, E son todas las variables de la red, cada término $P(y,e,w)$ es una instancia de la distribución conjunta. Además, la probabilidad $P(e)$ se puede calcular directamente sumando la conjunta de la siguiente forma:

$$P(e) = \sum_{y} P(y,e)$$

Finalmente, en las fórmulas anteriores, podemos dividir cada $P(y,e)$ por $P(e)$ y obtener la probabilidad condicionada $P(y|e)$.

5.2. Análisis de complejidad

En principio, se puede utilizar un modelo gráfico para responder a todos los tipos de consulta. Simplemente generando la distribución conjunta y sumándola exhaustivamente. Sin embargo, este enfoque del problema de la inferencia no es muy satisfactorio, ya que nos

lleva a la explosión exponencial de la distribución conjunta.

Por desgracia, el problema de la inferencia en modelos gráficos es NP-duro, y además, la inferencia aproximada es también NP-duro.

5.3. Eliminación de variables

En este trabajo, se va a explicar un algoritmo para hacer inferencia en los modelos gráficos probabilísticos llamado eliminación de variables. Este algoritmo es válido tanto para RBs como para redes de Markov. Cabe destacar que existen más algoritmos para el problema de inferencia.

Para comenzar, se necesita introducir el concepto de factor, concepto indispensable en el algoritmo de eliminación de variables.

Definición 5.1. Sea D un conjunto de variables aleatorias. Definimos un **factor** ϕ como una función de Val(D) a $\mathbb{R}$. Un factor es no negativo si todas sus entradas son no negativas. El conjunto de variables D es llamado el marco o *scope* del factor y es denotado como $Scope[\phi]$.

5.3.1. Marginalización de factores

Una de las operaciones claves que se hacen para calcular la probabilidad de un subconjunto de variables es la marginalización de variables de la distribución.

Definición 5.2. Sea X un conjunto de variables y sea $Y \notin X$ una variable. Sea $\phi(X, Y)$ un factor. Definimos la **marginalización** del factor de Y en ϕ, denotada por $\sum_Y \phi$, como un factor ψ sobre X tal que:

$$\psi(X) = \sum_Y \phi(X, Y)$$

A esta operación también se le llama sumar Y en ψ; ya que se suman las imágenes de Y, como se ve en la Figura 5.1.

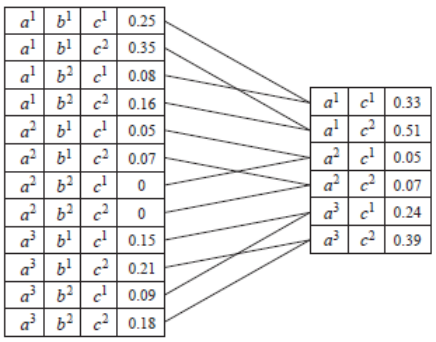

Figura 5.1.: ejemplo de marginalización de factores: sumar B

Definición 5.3. Sean los factores $\phi_1 : Val(A) \to \mathbb{R}$ y $\phi_2 : Val(B) \to \mathbb{R}$. Se define el **producto de factores** como el producto en $\mathbb{R}$ de las imágenes.

$$\phi_1 \cdot \phi_2 = \phi_1(a) \cdot \phi_2(b) \text{ con } a \in Val(A), b \in Val(B)$$

La marginalización de factores tiene tres propiedades importantes que se aplican en el algoritmo de eliminación de variables.

1. El producto es **conmutativo**; es decir: $\phi_1 \cdot \phi_2 = \phi_2 \cdot \phi_1$ o lo que es lo mismo $\sum_X \sum_Y \phi = \sum_Y \sum_X \phi$.
2. El producto es **asociativo**, es decir: $(\phi_1 \cdot \phi_2) \cdot \phi_3 = \phi_1 \cdot (\phi_2 \cdot \phi_3)$.
3. Si $X \notin Scope[\phi_1]$, entonces se da que $\sum_X (\phi_1 \cdot \phi_2) = \phi_1 \cdot \sum_X \phi_2$.

5.3.2. Algoritmo de eliminación de variables

Para facilitar la comprensión del algoritmo, se va a comenzar con un ejemplo simple. Se tiene la red $A \to B \to C \to D$. Con lo expuesto en el apartado anterior se puede escribir la distribución conjunta como un producto de factores.

$$P(A, B, C, D) = \phi_A \cdot \phi_B \cdot \phi_C \cdot \phi_D$$

Por otro lado, aplicando la propiedad 3 de los factores, se tiene que la distribución marginal sobre D es:

$$\begin{aligned} P(D) &= \sum_C \sum_B \sum_A P(A, B, C, D) \\ &= \sum_C \sum_B \sum_A \phi_A \cdot \phi_B \cdot \phi_C \cdot \phi_D \\ &= \sum_C \sum_B \phi_C \cdot \phi_D \cdot (\sum_A \phi_A \cdot \phi_B) \\ &= \sum_C \phi_D \cdot (\sum_B \phi_C \cdot (\sum_A \phi_A \cdot \phi_B)) \end{aligned}$$

A este paso se le llama la tarea de inferencia **suma-producto**. En general, cuando se quiere marginalizar Z de un conjunto de factores Φ, se denota de la siguiente forma:

$$\sum_Z \prod_{\phi \in \Phi} \phi$$

Una instanciación simple de este algoritmo es el procedimiento llamado eliminación de variables suma-producto (EV), que se muestra en el algoritmo 2.

Algoritmo 1: Suma-producto eliminar una variable

Datos:
Φ, // Conjunto de factores
Z // Variable a eliminar

1 $\Phi' \leftarrow \{\phi \in \Phi : Z \in Scope[\phi]\}$
2 $\Phi'' \leftarrow \Phi - \Phi'$
3 $\psi \leftarrow \prod_{\phi \in \Phi'} \phi$
4 $\tau \leftarrow \sum_Z \psi$
5 **devolver** $\Phi'' \cup \{\tau\}$

Algoritmo 2: Suma-producto EV

Datos:
Φ, // Conjunto de factores
Z, // Conjunto de variables a eliminar
$\prec$ //Orden en Z

Sea $Z_i, ..., Z_k$ un orden de **Z** tal que $Z_i \prec Z_j$ si y solo si $i < j$
para $i = 1,...,k$ **hacer**
  $\Phi \leftarrow$ Sum-Product-Eliminate-Var(Φ, Z_i)
fin
$\phi^* \leftarrow \prod_{\phi \in \Phi} \phi$
devolver ϕ^*

Con este algoritmo se puede conseguir calcular la distribución de probabilidad de una variable Y ($P_{\mathcal{B}}(Y)$) en una RB $\mathcal{B}$. Simplemente tomando todas las DPCs:

$$\Phi = \{\phi_{X_i}\}_{i=1}^{n}$$

donde $\phi_{X_i} = P(X_i|Pa_{X_i})$. En este momento, aplicando el algoritmo de eliminación de variables (Algoritmo 2) al conjunto $Z = \mathcal{X} - Y$ (elimando así las variables no demandadas) se consigue la distribución de la variables demandadas Y.

Para producir este efecto, el algoritmo suma-producto EV (2) elimina del conjunto de factores Φ, las variables del conjunto Z una a una mediante el algoritmo 1. El algoritmo 1, elimina una variable Z de un conjunto de factores Φ. Para ello, divide el conjunto Φ en dos: aquellos que tienen a Z en su *Scope* (Φ') y aquellos que no (Φ''). Una vez hecha la división de Φ, hace la tarea suma-producto sobre el conjunto Φ' para marginalizar Z y devuelve la marginalización junto con Φ''.

Una vez se han eliminado todas las variables, se multiplican los factores del conjunto Φ, obtieniendo el factor producto resultante con las variables no demandadas marginalizadas.

6. Aprendizaje

> En este capítulo se va a formalizar la idea de aprendizaje, definiendo los objetivos y reformulando el problema como una tarea de optimización.

Los conceptos que se definen en el capítulo se han basado en el libro *Probabilistic Graphical Models* [Koller and Friedman, 2009].

6.1. Motivación

Hasta ahora, siempre se ha empezado el razonamiento asumiendo tener el modelo gráfico probabilístico, tanto la estructura como los parámetros. Pero, no es siempre así. De hecho, la aplicación práctica que se desarrollará en la segunda parte no tiene conocimiento experto y por tanto no hay posibilidad de deducir ningún parámetro ni estructura a priori. En otras ocasiones, no hay expertos disponibles o bien se aprende de forma automática un modelo que luego el experto validará.

En estos casos, y bajo la suposición de que se tendrán suficientes datos del problema que se trate, se han desarrollado métodos de aprendizaje con una sólida base teórica.

6.2. Formalización de la idea de aprendizaje

Se asume, que el dominio del problema tiene una distribución subyacente $\boldsymbol{P^*}$, la cual está inducida por algún modelo denotado por $\boldsymbol{\mathcal{M}^*(\mathcal{K}^*, \theta^*)}$. Al conjunto de datos se le denota como $\boldsymbol{\mathcal{D} = \{d[1], ..., d[M]\}}$. Además, se asume que las instancias de este conjunto de datos son mutuamente independientes y tienen la misma distribución de probabilidad; es decir, que las instancias son independientes e idénticamente distribuidas (IID). También se necesita denotar por $\boldsymbol{\tilde{\mathcal{M}}}$ a una familia de modelos y $\boldsymbol{P_{\tilde{\mathcal{M}}}}$ o $\boldsymbol{\tilde{P}}$ a su distribución.

Por tanto, nuestro objetivo será que la distribución de la familia $\tilde{P}$ se aproxime lo máximo posible a P^*. Por lo que se tiene que seleccionar un modelo como la mejor aproximación posible de $\mathcal{M}^*$; el concepto de mejor, dependerá de el parámetro que se estime.

6.2.1. Estimación de la densidad

La razón más común para aprender un modelo gráfico es utilizarlo para alguna tarea de inferencia que se desee realizar. En este apartado el objetivo de aprendizaje será estimar la densidad; es decir, construir un modelo $\tilde{\mathcal{M}}$ tal que la distribución $\tilde{P}$ se aproxime lo mejor posible a P^*. Para esto, se define el concepto de entropía como una forma de cuantificar la incertidumbre. Esta definición nos lleva a definir la entropía relativa que se usará para medir la distancia entre las distribuciones $\tilde{P}$ y P^*. La definición de entropía siguiente, se basa en el libro *Bayesian networks and influence diagrams* [Kjaerulff and Madsen, 2008].

Definición 6.1. Sea $P(X)$ una distribución sobre una variable aleatoria X. La **entropía** de X se define como:

$$H_p(X) = -\mathbb{E}_p[log(P(X))] = -\sum_x P(x)log(P(X)) \geq 0$$

donde $\mathbb{E}_p$ es la esperanza matemática de la distribución P.

Definición 6.2. Sea $P(X_1, ..., X_n)$ una distribución sobre las variables aleatorias $X_1, ..., X_n$. La **entropía conjunta** de $X_1, ..., X_n$ se define como:

$$H_p(X_1, ..., X_n) = -\mathbb{E}_p[log(P(X_1, ..., X_n))]$$

Definición 6.3. Sea $P(X_1, ..., X_n)$ y $Q(X_1, ..., X_n)$ distribuciones sobre las variables aleatorias $X_1, ..., X_n$. La **entropía relativa** de P y Q se define como:

$$\mathbb{D}(P(X_1, ..., X_n)|Q(X_1, ..., X_n)) = \mathbb{E}_p[log(\frac{P(X_1, ..., X_n)}{Q(X_1, ..., X_n)})]$$

también llamada **divergencia de *Kullback–Leibler***.

Ahora, se puede definir la distancia entre distribuciones. Sea $\epsilon = (x_1, ..., x_n)$ una instancia de $(X_1, ..., X_n)$, se denota la **entropía relativa** sobre $\tilde{P}$ y P^* evaluada en ϵ como:

$$\mathbb{D}(P^*|\tilde{P}) = \mathbb{E}_{\epsilon \sim P^*}[log\frac{P^*(\epsilon)}{\tilde{P}(\epsilon)}]$$

Donde $\mathbb{E}_{\epsilon \sim P^*}$ es la esperanza de una instancia ϵ muestreada por una distribución P^*. Con esta expresión, se consigue evaluar la distancia entre $\tilde{P}$ y P^*. Esta distancia, cobra importancia con la siguiente proposición donde se obtiene una métrica para evaluar la bondad de $\tilde{P}$.

Proposición 6.1. *Para cualquier distribución P, P′ sobre $\mathcal{X}$:*

$$\mathbb{D}(P|P') = -H_p(\mathcal{X}) - \mathbb{E}_{\epsilon \sim P}[log(P'(\epsilon))]$$

Demostración.

$$\begin{aligned}
\mathbb{D}(P|P') &= \mathbb{E}_{\epsilon \sim P}[log\frac{P(\epsilon)}{P'(\epsilon)}] \\
&= \mathbb{E}_{\epsilon \sim P}[log(P(\epsilon)) - log(P'(\epsilon))] \\
&= \mathbb{E}_{\epsilon \sim P}[log(P(\epsilon))] - \mathbb{E}_{\epsilon \sim P}[log(P'(\epsilon))] \\
&= -H_p(\mathcal{X}) - \mathbb{E}_{\epsilon \sim P}[log(P'(\epsilon))]
\end{aligned}$$

La proposición anterior aporta la posibilidad de evaluar la distribución $\tilde{P}$ sin necesidad de tener P^*. Además, si se centra la métrica de evaluación en el segundo término $\mathbb{E}_{\epsilon \sim P}[log(P'(\epsilon))]$, haciéndolo lo más grande posible, se consigue disminuir esta distancia.

Intuitivamente, la disminución de la distancia entre las distribuciones P^* y $\tilde{P}$, se puede interpretar mediante la fórmula como el aumento de la probabilidad que el modelo da a los puntos muestreados. Para reflejar este concepto se define $\boldsymbol{P(\mathcal{D} : \mathcal{M})}$ como la probabilidad de los datos dado un modelo. En la práctica se usa el logaritmo de esta probabilidad.

Además, se necesita definir el concepto inverso. Es decir, la pérdida que se produce al muestrear un punto de P^* con la distribución $\tilde{P}$. Para ello, se define **la función de pérdida**, $\boldsymbol{loss(\epsilon : \mathcal{M})}$, como el error que tiene el modelo $\mathcal{M}$ sobre una instancia ϵ, en la práctica se usa el logaritmo $log\ loss(\epsilon : \mathcal{M})$. Esta definición implica que el objetivo sea encontrar un modelo $\mathcal{M}$ que minimice el **error esperado o riesgo** de la distribución P^*:

$$\mathbb{E}_{\epsilon \sim P^*}[loss(\epsilon : \mathcal{M})]$$

En general, P^* es desconocida. Sin embargo, podemos aproximarla calculando el **riesgo empírico medio** sobre el conjunto $\mathcal{D}$ de instancias de la distribución P^*:

$$\mathbb{E}_{\mathcal{D}}[loss(\epsilon : \mathcal{M})] = \frac{1}{\mathcal{D}} \sum_{\epsilon \in \mathcal{D}} loss(\epsilon : \mathcal{M})$$

El riesgo empírico medio puede utilizarse como métrica para evaluar la calidad de un modelo concreto. Además de como un factor para seleccionar un modelo entre una clase determinada dado un conjunto de entrenamiento $\mathcal{D}$.

6.2.2. Predicción

En la discusión anterior, se asume que el objetivo sería utilizar el aprendizaje para realizar inferencia probabilística. Se llega a la conclusión de que el objetivo es aproximar la distribución global P^* lo mejor posible. Sin embargo, con esto sólo se consigue medir la capacidad para evaluar la probabilidad de una instancia ϵ. En realidad, el modelo puede utilizarse para responder a toda una serie de consultas de la forma $P(Y|X)$. Es decir, el modelo puede responder a preguntas sobre la probabilidad de una variable Y dada un conjunto de variables predictoras X.

Si la variable Y fuese categórica, se estaría ante un problema de clasificación. Para obtener la categoría resultante de la predicción del modelo, haría falta establecer algún criterio para elegir el valor predicho. Para ello, se define $\boldsymbol{h_{\tilde{P}}(x)}$ cómo el valor predicho con mayor probabilidad.

Para poder estimar los parámetros en base al problema de predicción, se necesitan funciones de pérdida que evaluen la bondad del modelo frente a las predicciones realizadas. Para realizar esta tarea, se pueden definir dos funciones de pérdida como el error de clasificación o el criterio de la probabilidad condicionada.

Definición 6.4. El **error de clasificación** o error binario o/1, se define como la media de errores cometidos en la clasificación de la distribución. Es decir:

$$\mathbb{E}_{(x,y)\sim\tilde{P}}[h_{\tilde{P}}(x) \neq y] \text{ , } \forall \text{ } (x,y) \text{ pares muestreados de } \tilde{P}$$

El error de clasificación, como indica el nombre, solo se puede usar para problemas en los que la variable a predecir sea categórica.

Definición 6.5. El **criterio de la probabilidad condicionada** se define como:

$$\mathbb{E}_{(x,y)\sim\tilde{P}}[log\tilde{P}(y|x)] \text{ , } \forall \text{ } (x,y) \text{ pares muestreados de } \tilde{P}$$

El criterio de probabilidad condicionada evalúa el grado en que el modelo aprendido es capaz de predecir los datos generados a partir de la distribución.

6.2.3. Extracción de conocimiento

Finalmente, otra motivación muy diferente para aprender un modelo para la distribución es querer descubrir conocimiento sobre la misma. El modelo se puede traducir en conocimiento acerca de las dependencias directas e indirectas, la naturaleza de las dependencias, etc. Por ejemplo, en un determinado dominio, puede ser de gran interés conocer que parámetro tiene mayor importancia.

Con esta motivación, lo que buscamos es que $\tilde{\mathcal{M}}$ sea lo más parecido posible a $\mathcal{M}^*$, pasando a un segundo plano la distribución.

6.3. Optimización

En el apartado anterior se ha hablado de las diferentes formas para evaluar el modelo aprendido. Se ha definido un criterio numérico que se quiere optimizar, la función de pérdida. Por tanto, la tarea de aprendizaje puede verse desde una perspectiva de optimización. De hecho, se tiene un espacio de modelos y una función objetivo, por lo que, se cumplen las necesidades de la definición de un problema de optimización.

6.3.1. Riesgo empírico y sobreajuste

Se considera la tarea de construir un modelo que optimice la esperanza de una función de pérdida $\mathbb{E}_{\epsilon\sim\tilde{P}}[loss(\epsilon : \mathcal{M})]$. Igual que antes, normalmente no se sabe nada sobre P^*, pero, como se dijo, se puede usar el conjunto $\mathcal{D}$ muestreado de P^* para hacer una estimación empírica de esta esperanza.

Definición 6.6. Dado el conjunto $\mathcal{D} = \{\epsilon[1], ..., \epsilon[M]\}$ muestreado de P^*, se define la distribución empírica $\hat{P}_\mathcal{D}$ como:

$$\hat{P}_\mathcal{D} = \frac{1}{M}\sum_m I\{\epsilon[m] \in A\}$$

Donde la función I es la función característica que asigna un 1 si el elemento pertenece al conjunto y un 0 si no. Intuitivamente, la expresión representa la probabilidad del evento A como la fracción de las instancias del conjunto que satisfacen el evento.

Teorema 6.1. *Sea $\epsilon[1], \epsilon[2], ...$ una secuencia de instancias IID de $P^*(\mathcal{X})$, y sea $\mathcal{D}_M = \langle\epsilon[1], ..., \epsilon[M]\rangle$, entonces:*

$$\lim_{M\to\infty} \hat{P}_{\mathcal{D}_M}(A) = P^*(A)$$

Este teorema es muy importante, puesto que para un conjunto suficientemente grande, $\hat{P}_\mathcal{D}$ estará significativamente cerca de la distribución P^*. Por tanto, se puede usar la distribución empírica como la mejor aproximación a P^* e intentar minimizar la función de pérdida sobre $\hat{P}_\mathcal{D}$. Por desgracia, surgen muchos problemas.

Por ejemplo, si se usa la función de pérdida logarítmica ($logloss(\mathcal{D} : \mathcal{M})$) como objetivo. Está claro que la distribución que maximiza la función de pérdida logarítmica es la distribución empírica $\hat{P}_\mathcal{D}$. Esto provoca sobreajuste al conjunto de entrenamiento $\mathcal{D}$, provocando que la distribución no generalice y no consiguiendo nuestro objetivo, que es realizar consultas

acertadas sobre instancias de fuera del conjunto de entrenamiento.

6.4. Tareas del aprendizaje

Finalmente, para el proceso de aprendizaje, se debe proporcionar al algoritmo conocimiento a priori o restricciones sobre el modelo $\tilde{\mathcal{M}}$; además de un conjunto de datos para el entrenamiento con instancias IID.

En cuanto a las hipótesis sobre el modelo, se debe decidir si se aporta la estructura y se aprenden los parámetros, si se aprenden ambos o incluso si se conocen o no las variables aleatorias de la distribución objetivo. En la aplicación a la base de datos genética de la segunda parte del trabajo se aprenderá primero la estructura y luego se estimarán los parámetros de la distribución de probabilidad. Ambas tareas se estudiarán en el capítulo 7.

7. Estimación del modelo

En este capítulo se expone cómo estimar un modelo; paso final del proceso de aprendizaje estudiado en el capítulo 6. Para realizar esto, se divide en dos secciones, una para la estimación de la estructura de una RB sin hipótesis previas sobre la misma. Y otra para la estimación paramétrica, teniendo en cuenta que se tiene la estructura de la RB.

7.1. Estimación de la estructura

Como primer paso en la estimación del modelo se va a estimar la estructura de una RB en base a unos datos. Para ello, se van a definir una serie de algoritmos. Lo descrito en esta sección se basa en el texto sobre algoritmos para aprender la estructura del artículo *Using Bayesian networks to discover relations between genes, environment, and disease* [Su et al., 2013].

Para aprender la estructura hay tres metodologías; una basada en restricciones (*constraint-based*); otra basada en la evaluación numérica de la estructura, llamada basada en métricas; (*score-based*) y una híbrida.

7.1.1. Basada en restricciones

Los métodos basados en restricciones se centran en identificar las relaciones de independencia condicional entre las variables utilizando los datos observados. Normalmente, para estudiar la independencia entre dos variables se usan test de independencia como el test χ^2. Los tests aportan únicamente la información sobre la existencia de dependencia, no sobre la dirección de la dependencia.

A continuación, se añaden direcciones a las aristas entre los nodos según el criterio de d-separación. Todas las relaciones de d-separación entre los nodos de un grafo implican relaciones de independencia condicional entre las variables correspondientes. Sin embargo, no siempre se da el caso de que un conjunto concreto de independencias condicionales especifique un único grafo dirigido. Esto se debe a que hay ciertas aristas en las que la dirección de la independencia condicionada no es suficientemente apoyada por el conjunto de datos. Por esto, a veces se obtiene como salida un grafo parcialmente dirigido, el cuál se puede interpretar como el representante de los grafos aprendidos en base al conjunto de entrenamiento.

Para los métodos basados en restricciones se va a estudiar dos algoritmos: *Grow-Shrink* (GS) y *Incremental Association Markov Blanket* (IAMB).

7.1.1.1. *Grow-Shrink* (GS)

Grow-Shrink (GS) es un algoritmo basado en restricciones que se basa en el concepto del manto de Markov. El **manto de Markov** (*Markov blanket*) de un nodo en una RB consiste en sus padres, hijos y los padres de sus hijos. Estos representan todas las variables influidas por

el nodo. La explicación del algoritmo GS está basada en el artículo *Learning Bayesian Network Model Structure from Data* [Margaritis, 2003].

El algoritmo Grow-Shrink tiene dos fases diferenciadas en la creación del manto de Markov de una variable, una fase de crecimiento (*grow*) y otra de reducción (*shrink*). El algoritmo GS comienza con una variable X y un conjunto de variables S vacío. La **fase de crecimiento** añade a S las variables que sean dependientes de X condicionadas a las variables que se encuentran actualmente en S. La **fase de reducción** elimina las variables que se vuelven independientes de X, condicionadas a los miembros actuales de S. Para conseguir la información sobre al independencia, el algoritmo GS usa el test χ^2.

Algoritmo 3: Calcular el Manto de Markov

Datos:
$\mathcal{U}$, // Conjunto de variables aleatorias
$B(X)$, // manto de Markov de X
1 $S \leftarrow \emptyset$
2 Mientras $\exists Y \in \mathcal{U}$ tal que Y depende de X dado S; hacer $S \leftarrow S \cup \{Y\}$.
3 Mientras $\exists Y \in S$ tal que Y **no** depende de X dado $S - \{Y\}$; hacer $S \leftarrow S - \{Y\}$.
4 $B(X) \leftarrow S$

Las variables que quedan en $B(X)$ después de ambas fases representan entonces una estimación del manto de Markov (*Markov blanket*) de X, junto con X. Una vez se tiene claro como se calcula el manto de Markov para una variable, se explica el algoritmo GS en el pseudocódigo siguiente.

Algoritmo 4: Grow-Shrink (GS)

Datos:
$\mathcal{U}$, // Conjunto de variables aleatorias
$B(X)$, // manto de Markov de X
$N(X)$, // vecinos de X
1 **1. [Calcular el Manto de Markov [3]]**
2 Para todo $\boldsymbol{X \in \mathcal{U}}$, calcular $\boldsymbol{B(X)}$
3 **2. [Calcular la estructura del grafo]**
4 Para todo $\boldsymbol{X \in \mathcal{U}}$ e $\boldsymbol{Y \in B(X)}$, determina si $\boldsymbol{Y}$ es un vecino directo de $\boldsymbol{X}$ si $\boldsymbol{X}$ e $\boldsymbol{Y}$ son dependientes dado $\boldsymbol{S}$ para todo $\boldsymbol{S \subset T}$, donde $\boldsymbol{T}$ es el conjunto más pequeño entre $\boldsymbol{B(X) - \{Y\}}$ y $\boldsymbol{B(Y) - \{X\}}$.
5 **3. [Orientación de los lados]**
6 Para todo $\boldsymbol{X \in \mathcal{U}}$ e $\boldsymbol{Y \in N(X)}$, la orientación es $\boldsymbol{Y \rightarrow X}$ si existe una variable $\boldsymbol{Z \in N(X) - N(Y) - \{Y\}}$ tal que $\boldsymbol{Y}$ y $\boldsymbol{Z}$ son dependientes dados $\boldsymbol{S \cup \{X\}}$ para todo $\boldsymbol{S \subset T}$, donde $\boldsymbol{T}$ es el conjunto más pequeño entre $\boldsymbol{B(Y) - \{X, Z\}}$ y $\boldsymbol{B(Z) - \{X, Y\}}$.
7 **4. [Eliminar ciclos]**
8 Mientras existan ciclos en el grafo, calcula el nodo que pertenezca al mayor número de ciclos y eliminalo del grafo. Añade este nodo al conjunto $\boldsymbol{R}$.
9 **5. [Revertir R]**
10 Inserta los nodos en el grafo en orden inverso de insercción en $\boldsymbol{R}$
11 **6. [Propagar las direcciones]**
12 Para todo $\boldsymbol{X \in \mathcal{U}}$ e $\boldsymbol{Y \in N(X)}$, si existe un camino orientado de $\boldsymbol{X}$ a $\boldsymbol{Y}$, se orienta $\boldsymbol{X \rightarrow Y}$

Como se muestra en el algoritmo 4. Lo primero que se hace en el algoritmo es calcular el manto de Markov de todos los nodos. Una vez se ha calculado; se determinan para cada nodo los vecinos directos que tiene, consiguiendo crear el esqueleto del grafo. Luego, se orientan las aristas usando la d-separación.

Una vez realizado estos pasos, nos encontramos ante un grafo que puede ser cíclico; por lo que el siguiente paso es eliminar los nodos que forman más ciclos. Una vez no quedan más ciclos, se introducen en el grafo en orden inverso. Por último, se propagan las orientaciones.

7.1.1.2. *Incremental Association Markov Blanket* (IAMB)

Incremental Association Markov Blanket (IAMB) es un algoritmo para la estimación del manto de Markov de una variable. Al igual que el algoritmo GS, trata de calcular el manto de Markov (*Markov blanket*) con dos fases, una de **incorporación** y otra de **eliminación**. Este algoritmo, se va a basar en el artículo *Algorithms for Large Scale Markov Blanket Discovery* [Tsamardinos et al., 2003].

Algoritmo 5: IAMB

Datos:
I, // criterio de independencia
f, // función heurística para el orden de recorrer los nodos
T, // variable a estimar el manto de Markov

```
1  CMB ← ∅
2  //FASE INCORPORACIÓN
3  mientras CMB cambie hacer
4  |  encuentra una variable X en V − CMB − {T} que maximice f(X; T|CMB)
5  |  si no I(X; T|CMB) (se produce dependencia) entonces
6  |  |  CMB ← CMB ∪ X
7  |  fin
8  fin
9  //FASE ELIMINACIÓN
10 Elimina de CMB todas las variables X, para las que I(X; T|CMB − {X}) (se
     produzca independencia)
```

IAMB utiliza una función heurística dinámica para determinar el orden de las variables, $f(X;T|CMB)$. Esta función se calcula con la información mutua entre X y T, condicionada a las variables del conjunto del manto de Markov candidato, CMB. Además, IAMB no fija el criterio de independencia.

Finalmente, para transformar esta información en una RB usa un algoritmo similar al 4. Usando la d-separación para dirigir los lados del grafo y propagar las dependencias.

7.1.2. Aprendizaje basado en métricas

Estos métodos se centran en puntuar las estructuras de RBs para medir la bondad de la misma con respecto a los datos. Los algoritmos que usan esta metodología tratan de maximizar la puntuación o métrica realizando una búsqueda sobre el espacio de los posibles grafos y devolviendo aquella estructura que más puntuación tenga.

Para calcular la puntuación existen varias métricas. Para facilitar la notación se van a denotar por $P(\mathcal{D}|\hat{\theta},\mathcal{G})$, a la probabilidad del conjunto de datos $\mathcal{D}$ con el EMV explicado en la sección 7.2.1 y la estructura de RB $\mathcal{G}$. Las métricas que se van a usar se basan en el artículo *A new characterization of equivalent Bayesian network structures* [Chickering, 1995].

- ***Bayesian Information Criterion* (BIC)**.

  $$BIC = log(P(\mathcal{D}|\hat{\theta},\mathcal{G})) - \frac{n_p}{2} log(N)$$

  donde N es el tamaño del conjunto $\mathcal{D}$ y n_p el tamaño del conjunto de parámetros que tiene la estructura; es decir, el número de parámetros que luego se necesitará estimar. Esta métrica penaliza las estructuras complejas, con número de parámetros muy grandes.

- ***Akaike Information Criterion*(AIC)**.

  $$AIC = log(P(\mathcal{D}|\hat{\theta},\mathcal{G})) - n_p$$

  donde n_p el tamaño del conjunto de parámetros. Al contrario que la métrica BIC, AIC no penaliza la complejidad; si no que hace énfasis en la proximidad a los datos.

- ***Log-likelihood***. La cual se estudiará en profundidad en este capítulo en la sección 7.2.1. Con esta métrica, la estructura se centra en aproximar la distribución de los datos; sin importar la complejidad de la estructura.

- ***K2 score***. Esta métrica se calcula sobre cada una de las variables. Luego, se suman los resultados de cada variable para obtener la métrica K2 global de la estructura.

  $$log(K2(X_i)) = \sum_{j=1}^{q_i} (ln(\frac{(r_i-1)!}{(N_{ij}+r_i-1)!}) + \sum_{k=1}^{r_i} ln(N_{ijk}!)$$

  donde r_i es el número de posibles valores de la variable X_i; N_{ijk} representa el número de casos del conjunto de datos en el cual la variable X_i toma su k-ésimo valor ($k = 1, 2, ..., r_i$). El conjunto de padres es instanciado como la j-ésima combinación de valores ($j = 1, 2, ..., q_i$); donde q_i es el número de padres. Y $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$ es el número de casos del conjunto de datos del padre j-ésimo de la variable X_i.

Los métodos basados en métricas, son a veces más usados que los basados en restricciones porque el grafo resultante es siempre un grafo dirigido acíclico. El algoritmo que se va a estudiar que usa esta metodología es el *Hill-Climbing* (HC).

7.1.2.1. *Hill-Climbing* (HC)

Hill-Climbing (HC) es un ejemplo de algoritmo basado en métricas. La búsqueda sobre las estructuras comienza a partir de un grafo vacío, completamente conectado o aleatorio. Alternativamente, el grafo inicial puede ser elegido mediante conocimiento experto. El bucle principal consiste en iterar sobre cada posible incorporación, eliminación o inversión de una arista del grafo actual. El cambio que más aumenta la puntuación se incorpora y se actualiza el grafo actual. El proceso se repite hasta que no existe cambio que aumente la puntuación. El psudocódigo proporcionado se basa en el libro *Bayesian networks in r* [Nagarajan et al., 2013].

Algoritmo 6: HC

Datos:
$\mathcal{G}$, // estructura de RB
V, // conjunto de variables aleatorias
Elige una estructura inicial $\mathcal{G}$ sobre V, bajo algún criterio.
Calcula la métrica de $\mathcal{G}$ elegida, $metrica(\mathcal{G})$
mientras *La puntuación aumente* **hacer**
  para cada *eliminación, incorporación o inversión de una arista que no genere un ciclo* **hacer**
    Calcula la métrica del grafo modificado $\mathcal{G}^*$, $metrica(\mathcal{G}^*)$
    si $metrica(\mathcal{G}^*) > metrica(\mathcal{G})$ **entonces**
      $\mathcal{G} = \mathcal{G}^*$ $metrica(\mathcal{G}) = metrica(\mathcal{G}^*)$
    fin
  fin
fin
devolver $\mathcal{G}$

7.1.3. Metodología híbrida

Por último, también se han desarrollado algoritmos híbridos que combinan las dos metodologías convencionales para maximizar sus ventajas. Normalmente, comienzan con un algoritmo basado en restricciones para encontrar el esqueleto de la red y luego emplean un método basado en métricas para identificar el mejor conjunto de orientaciones de las aristas. El algoritmo que se va a estudiar en esta sección es *Max-Min Hill-Climbing* (MMHC).

7.1.3.1. *Max-Min Hill-Climbing* (MMHC)

Primero, aprende el esqueleto de un grafo utilizando un algoritmo basado en restricciones llamado *Max-Min Parents and Children* (MMPC). Luego, se aplica HC para orientar las aristas.

MMPC consta de dos fases al igual que los otros algoritmos basados en restricciones. En la primera fase; para cada variable X, se van conectando variables a X con aristas de acuerdo a la heurística $Max - Min$ dinámica influenciada por el conjunto de variables conectadas. En la segunda fase, se eliminan las dependencias falsas mediante algún test de dependencia. Una vez realizado esto para todas las variables, se ha realizado una estimación de los

mantos de Markov y se puede construir el esqueleto.

Algoritmo 7: MMPC

Datos:
V, // conjunto de variables aleatorias S, // conjunto de variables conectadas a X
$Max - Min(X;Y|S)$ // función heurística

```
1  para cada variable X de V hacer
2  |  S ← ∅
3  |  //FASE 1
4  |  para cada variable Y de V elegida según Max − Min(X;Y|S) hacer
5  |  |  S ← S ∪ {Y}
6  |  fin
7  |  //FASE 2
8  |  para cada variable Y de S hacer
9  |  |  si un cierto test de dependencia no avala la relación entre X e Y, entonces
   |  |   S ← S − {Y}
10 |  fin
11 fin
```

La función $\boldsymbol{Max - Min(X;Y|S)}$ es el valor máximo de las mínimas asociaciones entre X e Y. Donde las asociaciones $Assoc(X;Y|T)$ es una estimación de la dependencia de X sobre Y dado el conjunto de variables T . Las mínimas asociaciones se define con la siguiente expresión.

$$Min - Assoc(X;Y|S) = \min_{T \subset S} Assoc(X;Y|T)$$

Por tanto, la función $Max - Min(X;Y|S)$ se define como el máximo de $Min - Assoc(X;Y|S)$.

Una vez se ha definido el esqueleto, se usa el algoritmo HC para dirigir las aristas del grafo; obteniéndose un grafo acíclico dirigido.

7.2. Estimación de los parámetros

Esta sección está basada en los conceptos definidos en el libro *Probabilistic Graphical Models* [Koller and Friedman, 2009] en el ámbito de la estimación paramétrica.

Una vez se ha aprendido la estructura de RB en la sección anterior; se va a estudiar el problema de la estimación de los parámetros. Los parámetros de una RB son los valores de las relaciones de independencia existentes en la red. Para tratar este problema hay dos enfoques: la estimación del estadístico máximo verosímil (EMV) y el enfoque bayesiano. Se va a estudiar el EMV.

7.2.1. Estadístico máximo verosímil (EMV)

Con el EMV se va a conseguir una forma de calcular los parámetros de la RB. Para comenzar el estudio del EMV se necesitan fijar algunos términos. Denotamos por $\boldsymbol{\theta}$ al vector de parámetros de la RB, $\boldsymbol{\Theta}$ al espacio donde toma valores y $\boldsymbol{\mathcal{D} = \{\epsilon[1], ..., \epsilon[M]\}}$ al conjunto de datos sobre los que se realiza el aprendizaje, el conjunto de entrenamiento. En primer lugar, se necesita conocer la definición de EMV.

Definición 7.1. Decimos que $\hat{\theta}$ es el **estadístico máximo verosímil (EMV)** cuando satisface:

$$L(\hat{\theta} : \mathcal{D}) = \max_{\theta \in \Theta} L(\theta : \mathcal{D})$$

siendo $L(\theta : \mathcal{D}) = \prod_{m \in 1,..,M} P(\epsilon[m] : \theta)$ la **función de verosimilitud** y $P(\epsilon[m] : \theta)$ la probabilidad de la instancia $\epsilon[m]$ bajo los parámetros θ.

Se dice que una función de verosimilitud tiene la propiedad de descomponibilidad si se descompone en factores de otras funciones de verosimilitud. Las RBs permiten reducir el problema de estimación de parámetros a un conjunto de problemas no relacionados; cada uno de los cuales puede abordarse utilizando el EMV. Antes de abordar la decomposición hay que definir un concepto previo.

Definición 7.2. Sea Z un conjunto de variables aleatorias, z alguna instancia de estas variables aleatorias y $\mathcal{D}$ un conjunto de datos. Denotamos como $M[z]$ al número de entradas en $\mathcal{D}$ que tienen $Z[m] = z$.

$$M[z] = \sum_{m=1}^{M} I\{Z[m] = z\}$$

donde $I\{Z[m] = z\}$ es la función cuyo resultado es 1 si se da la condición que aparece entre llaves y 0 si no.

7.2.1.1. Descomposición función de verosimilitud

Sea θ los parámetros que se quieren estimar en la RB $\mathcal{G}$ y sea $X = X_1, ..., X_n$ el conjunto de variables aleatorias representadas en $\mathcal{G}$. Gracias a la definición de función de verosimilitud y a las relaciones de independencia de $\mathcal{G}$, se puede reformular la función de verosimilitud en un producto de probabilidades condicionales.

$$
\begin{aligned}
L(\theta : \mathcal{D}) &= \prod_{m=1}^{M} P_{\mathcal{G}}(\epsilon[m] : \theta) \\
&= \prod_{m=1}^{M} \prod_{i=1}^{n} P(x_i[m]|pa_{X_i} : \theta) \\
&= \prod_{i=1}^{n} [\prod_{m=1}^{M} P(x_i[m]|pa_{X_i}[m] : \theta)]
\end{aligned}
$$

Con esta expresión, se puede observar que se ha descompuesto la función de verosimilitud en un conjunto de factores que son funciones de verosimilitud. Para clarificar, se usará la notación $\theta_{X_i|Pa_{X_i}}$ para indicar el subconjunto de parámetros que determinan la probabilidad $P(X_i|Pa_{X_i})$ en $\mathcal{G}$. Con esta notación, se puede reescribir la expresión anterior como un producto de funciones de verosimilitud.

$$L(\theta : \mathcal{D}) = \prod_{i=1}^{n} L_i(\theta_{X_i|Pa_{X_i}} : \mathcal{D})$$

donde la función de verosimilitud para X_i es:

$$L_i(\theta_{X_i|Pa_{X_i}} : \mathcal{D}) = \prod_{m=1}^{M} P(x_i[m]|pa_{X_i}[m] : \theta_{X_i|Pa_{X_i}})$$

Con estos cálculos simples se demuestra que la función de verosimilitud se descompone en un producto de factores, uno por DPC de la red. Esta propiedad es muy importante y es llamada **descomposición global**. De esta propiedad podemos derivar la siguiente proposición:

Proposición 7.1. *Sea $\mathcal{D}$ un conjunto de datos para $X_1, ..., X_n$ y $\mathcal{G}$ la estructura de RB sobre esas variables. Suponiendo que los parámetros $\theta_{X_i|Pa_{X_i}}$ y $\theta_{X_j|Pa_{X_j}}$ son disjuntos para todo $i \neq j$. Sea $\hat{\theta}_{X_j|Pa_{X_j}}$ los parámetros que maximizan $L_i(\theta_{X_i|Pa_{X_i}} : \mathcal{D})$. Entonces, $\hat{\theta} = \langle \hat{\theta}_{X_1|Pa_{X_1}}, ..., \hat{\theta}_{X_n|Pa_{X_n}} \rangle$ maximiza $L(\theta : \mathcal{D})$.*

De este resultado se deduce la posibibilidad de reducir el problema de maximización global de la función de verosimilitud del conjunto de variables aleatorias X de $\mathcal{G}$; a un problema de maximización local sobre cada una de las variables X_i de $\mathcal{G}$.

Una vez expuesta la idea de como estimar los parámetros de una RB, se van a estudiar dos casos concretos de representación de distribuciones de probabilidad: las **tablas DPCs** y las **redes Bayesianas gaussianas**. Para cada uno de ellos se calculará el EMV.

7.2.1.2. Tablas DPCs

En este apartado se va a desarrollar la teoría para la estimación paramétrica sobre la distribución de una RB en formato de tabla. Para ello, se va a calcular el EMV asociado a las tablas DPCs.

Sea X una variable y U sus padres. Se asume que son variables discretas. Si se representa la DPC $P(X|U)$ como una tabla, entonces, un parámetro $\theta_{x|u}$ para cada combinación de

$x \in Val(X)$ y $u \in Val(U)$ será la probabilidad de que se de x habiéndose dado u. Por tanto, se puede reescribir la función de verosimilitud para distribuciones en formato tabla.

$$L_X(\theta_{X|U} : \mathcal{D}) = \prod_{m=1}^{M} \theta_{x[m]|u[m]} \tag{7.1}$$

Si agrupamos todas las ocurrencias de $\theta_{x|u}$ en el producto de todas las intancias, se obtiene la función de verosimilitud local para distribuciones en formato tabla.

$$L_X(\theta_{X|U} : \mathcal{D}) = \prod_{u \in Val(U)} [\prod_{x \in Val(X)} \theta_{x|u}^{M[u,x]}]$$

donde $M[u, x]$ es el número de veces que $\epsilon[m] = x$ y $u[m] = u$ en $\mathcal{D}$. Por último, queda maximizar este término con la restricción de que para cada valor de los padres u de U, el parámetro esté bien definido. Es decir, que la suma de las probabilidades de x condicionado a los valores del padre U sume 1.

$$\sum_{u \in Val(U)} \theta_{x|u} = 1$$

Esta restricción implica que la elección del valor $\theta_{x|u}$ puede alterar el valor de $\theta_{x'|u}$. Sin embargo, la elección de los valores u es independiente entre sí. Por ello, se pueden maximizar cada factor del productorio con respecto a u de la ecuación de la descomposición local para distribuciones en formato tabla (7.1), por separado.

Por último, el libro *Probabilistic Graphical Models* [Koller and Friedman, 2009] aporta una demostración concluyendo que cada función de verosimilitud local es un tipo de multinomial y calculando el EMV a partir de este hecho. Se obtiene que el EMV para tablas DPCs es el siguiente.

$$\hat{\theta}_{x|u} = \frac{M[u, x]}{M[u]} \text{ , donde } M[u] = \sum_{x \in Val(X)} M[u, x]$$

7.2.1.3. Red Bayesiana Gaussiana

En este apartado se va a calcular el EMV sobre una variable para RBs gaussianas-lineales. Para esto, es necesario fijar la variable X y sus padres $U = \{U_1, ..., U_k\}$, asumiendo que la variable a estudiar sigue una DPC lineal gaussiana.

$$P(X|u) \sim \mathcal{N}(\beta_0 + \beta_1 u_1 + ... + \beta_k u_k; \sigma^2)$$

Los parámetros que presenta esta distribución son $\langle \beta_0, ..., \beta_k; \sigma \rangle$ y se van a denotar como $\theta_{X|U}$. Para el cálculo del EMV se usa el logaritmo de la función de verosimilitud para facilitar los cálculos.

$$logL_X(\theta_{X|U} : \mathcal{D}) = \sum_m [-\frac{1}{2} log(2\pi\sigma^2) - \frac{1}{2\sigma^2}(\beta_0 + \beta_1 u_1[m] + ... + \beta_k u_k[m] - x[m])^2]$$

Una vez se obtiene el logaritmo de la función de verosimilitud, se busca maximizarla para obtener el EMV. Por tanto, hay que calcular las derivadas parciales e igualarlas a cero.

Primero se va a hacer la derivada parcial con respecto a β_0.

$$\begin{aligned}\frac{\partial}{\partial\beta_0} log L_X(\theta_{X|U} : \mathcal{D}) &= \sum_m -\frac{1}{\sigma^2}(\beta_0 + \beta_1 u_1[m] + ... + \beta_k u_k[m] - x[m]) \\ &= -\frac{1}{\sigma^2}(M\beta_0 + \sum_m \beta_1 u_1[m] + ... + \sum_m \beta_k u_k[m] - \sum_m x[m])\end{aligned}$$

Igualando a cero y multiplicando por $\frac{\sigma^2}{M}$, se obtiene la siguiente expresión.

$$\beta_0 + \frac{1}{M}\sum_m \beta_1 u_1[m] + ... + \frac{1}{M}\sum_m \beta_k u_k[m] = \frac{1}{M}\sum_m x[m]$$

Cada uno de los términos $\frac{1}{M}\sum_m \beta_i u_i[m]$ con $i = 1, ..., k$ es la esperanza matemática en el conjunto $\mathcal{D}$ de la variable U_i.

$$\mathbb{E}_\mathcal{D}[X] = \beta_0 + \beta_1 \mathbb{E}_\mathcal{D}[U_1] + ... + \beta_k \mathbb{E}_\mathcal{D}[U_k] \tag{7.2}$$

Con esta expresión, se consigue escribir la media de la variable X en términos de las medias de sus padres. Por tanto, si se considera el gradiente con respecto a un parámetro β_i con $i \neq 0$ y aplicando las mismas operaciones se obtienen k expresiones de la siguiente forma.

$$\mathbb{E}_\mathcal{D}[X \cdot U_i] = \beta_0 \mathbb{E}_\mathcal{D}[U_i] + \beta_1 \mathbb{E}_\mathcal{D}[U_1 \cdot U_i] + ... + \beta_k \mathbb{E}_\mathcal{D}[U_k \cdot U_i] \text{ , } i \in \{1, ..., k\} \tag{7.3}$$

En este punto, se tienen $k+1$ ecuaciones lineales con $k+1$ parámetros desconocidos, pudiendo resolver y calcular los parámetros β_i con $i \in \{0, ..., k\}$. Por tanto, solo faltaría averiguar el parámetro σ. Para averiguar *sigma*, primero hay que restar a la ecuación (7.3) el valor $\mathbb{E}_\mathcal{D}[X]\mathbb{E}_\mathcal{D}[U_i]$; obteniendo la siguiente expresión.

$$\begin{aligned}&\mathbb{E}_\mathcal{D}[X \cdot U_i] - \mathbb{E}_\mathcal{D}[X]\mathbb{E}_\mathcal{D}[U_i] = \\ &\beta_0 \mathbb{E}_\mathcal{D}[U_i] + \beta_1 \mathbb{E}_\mathcal{D}[U_1 \cdot U_i] + ... + \beta_k \mathbb{E}_\mathcal{D}[U_k \cdot U_i] - \mathbb{E}_\mathcal{D}[X]\mathbb{E}_\mathcal{D}[U_i]\end{aligned} \tag{7.4}$$

En segundo lugar, se multiplica la ecuación (7.2) por $\mathbb{E}_\mathcal{D}[U_i]$, consiguiendo uno de los términos de la parte izquierda de la ecuación (7.4).

$$\mathbb{E}_\mathcal{D}[X]\mathbb{E}_\mathcal{D}[U_i] = \beta_0 \mathbb{E}_\mathcal{D}[U_i] + \beta_1 \mathbb{E}_\mathcal{D}[U_1]\mathbb{E}_\mathcal{D}[U_i] + ... + \beta_k \mathbb{E}_\mathcal{D}[U_k]\mathbb{E}_\mathcal{D}[U_i] \tag{7.5}$$

Luego, sustituyendo en la ecuación (7.4), la ecuación (7.5); se consigue la expresión siguiente.

$$\begin{aligned}&\mathbb{E}_\mathcal{D}[X \cdot U_i] - \mathbb{E}_\mathcal{D}[X] \cdot \mathbb{E}_\mathcal{D}[U_i] = \\ &\beta_1(\mathbb{E}_\mathcal{D}[U_i \cdot U_1] - \mathbb{E}_\mathcal{D}[U_1]\mathbb{E}_\mathcal{D}[U_i]) + ... + \beta_k(\mathbb{E}_\mathcal{D}[U_k \cdot U_i] - \mathbb{E}_\mathcal{D}[U_k]\mathbb{E}_\mathcal{D}[U_i])\end{aligned}$$

En este punto podemos sustituir por la covarianza en $\mathcal{D}$, ya que $\mathbb{E}_\mathcal{D}[X \cdot Y] - \mathbb{E}_\mathcal{D}[X] \cdot \mathbb{E}_\mathcal{D}[Y] = Cov_\mathcal{D}[X; Y]$.

$$Cov_\mathcal{D}[X; U_i] = \beta_1 Cov_\mathcal{D}[U_1; U_i] + ... + \beta_k Cov_\mathcal{D}[U_k; U_i]$$

Y finalmente, usando esta ecuación y el resultado de derivar la función de verosimilitud con respecto a σ e igualar a cero, se obtiene:

$$\sigma^2 = Cov_\mathcal{D}[X; X] - \sum_i \sum_j Cov_\mathcal{D}[U_i; U_j]$$

Por tanto, el EMV sobre una variable para RBs gaussianas-lineales sobre los parámetros $\langle \beta_0, ..., \beta_k; \sigma \rangle$ lo componen, las medias de X y U, además de la matriz de covarianzas de $\{X\} \cup U$.

Parte II.

Aplicación a Datos Genéticos

En esta segunda parte se aplica lo expuesto en la Parte 1 a una base de datos genética. Para ello, primero se revisan los estudios previos en el capítulo 8, se hará un análisis explicativo de la base de datos que se usa en el capítulo 9 y se hará una explicación del aprendizaje de la red Bayesiana sobre los datos explicados usando un cuaderno *Python* en el capítulo 10.

8. Aplicaciones de las Redes Bayesianas al análisis de datos genéticos

> En este capítulo vamos a estudiar artículos previos que analizan las relaciones entre los datos de una base de datos genética mediante redes bayesianas.

Para tener una visión más general a la hora de aprender una RB a partir de los datos genéticos que se van a estudiar, se han elegido los artículos *Using Bayesian Networks to Analyze Expression Data* [Friedman et al., 2000] y *Using Bayesian networks to discover relations between genes, environment, and disease* [Su et al., 2013].

8.1. *Using bayesian networks to analyze expression data*

En este artículo, los autores tratan de usar las redes Bayesianas para representar las dependencias estadísticas entre los genes. Los autores afirman que las RBs son muy atractivas por su capacidad para describir procesos estocásticos complejos y porque proporcionan metodologías claras para aprender de las observaciones ruidosas.

El artículo empieza mostrando cómo las redes Bayesianas pueden describir las interacciones entre genes. Luego, describe un método para recuperar las interacciones de los genes usando herramientas para el aprendizaje de las redes Bayesianas parecidas a las vistas en el capítulo 6. Y Finalmente, aplica este método a las mediciones del ciclo celular de *S. cerevisiae* [Spellman et al., 1998].

Este novedoso intento en el estudio de las relaciones entre los genes surge del avance tecnológico que proporcionó el microarray de ADN. Con él se puede estudiar la expresión de miles de genes simultáneamente, lo que ayudó a que estudios como este tuvieran la oportunidad de surgir.

La estrategia que siguen los autores para estudiar las relaciones entre los genes mediante RBs es definir una variable aleatoria por cada gen del microarray de ADN. Además, se podrán añadir más variables aleatorias que pudiesen tener efecto en los resultados, como son: las condiciones experimentales, indicadores temporales etc. A continuación, como ya se ha estudiado, tuvieron que elegir un modelo probabilístico local para implementar la red Bayesiana teniendo en cuenta que la variable de expresión de un gen es continua. Estudiaron dos modelos:

- **Modelo Multinomial**. Modelo discreto, para el cual discretizaron los datos en : *under-expressed* (-1), *normal* (0), and *over-expressed* (1). Dependiendo del ratio de expresión de los genes con respecto a un valor de control dado.
- **Modelo Gaussiano-lineal**. Al ser un modelo continuo no se tuvieron que aplicar

transformaciones a los datos, pues las intensidades de los genes son variables continuas.

Los autores dan como justificación a que se estudien dos modelos, pues ambos tienen desventajas sobre los datos. Con el modelo multinomial se pierde información al discretizar los valores. Y con el modelo gaussiano-lineal solo se pueden obtener dependencias cercanas a la linealidad.

Finalmente, aprenden de los datos de forma similar a como se ha plasmado en la primera parte sobre los datos de *S. cerevisiae* [Spellman et al., 1998]. Este conjunto de datos contiene 76 mediciones de expresión génica de los niveles de ARNm de 6177 *ORF de S. cerevisiae*. Los experimentos miden seis series temporales bajo diferentes métodos de sincronización del ciclo celular. En la base de datos que estudian, se identificaron 800 genes cuya expresión variaba a lo largo de las diferentes etapas del ciclo celular.

Tratan todas las muestras como independientes y toman una variable aleatoria por gen, aparte de una para controlar el ciclo celular. Tras proceder al aprendizaje de la red, los autores comentan que las características aprendidas muestran que podemos recuperar una estructura incluso a partir de conjuntos de datos pequeños. Y señalan que el algoritmo de aprendizaje no utiliza conocimientos biológicos previos ni restricciones. Todas las redes y relaciones aprendidas se basan únicamente en la información transmitida en las propias mediciones de los datos.

Finalmente, obtienen la siguiente red Bayesiana (obtenida de las diapositivas explicativas del artículo [Friedman et al., 1999]).

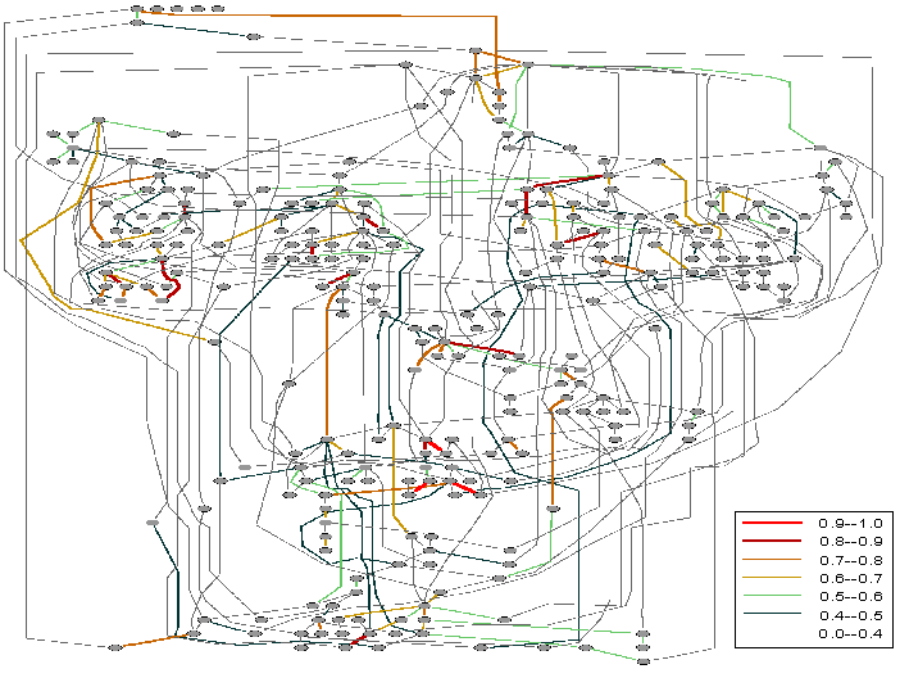

Figura 8.1.: Red Bayesiana

Además, en el artículo se proporciona un grafo que muestra de forma local las relaciones de independencia del gen SVS1.

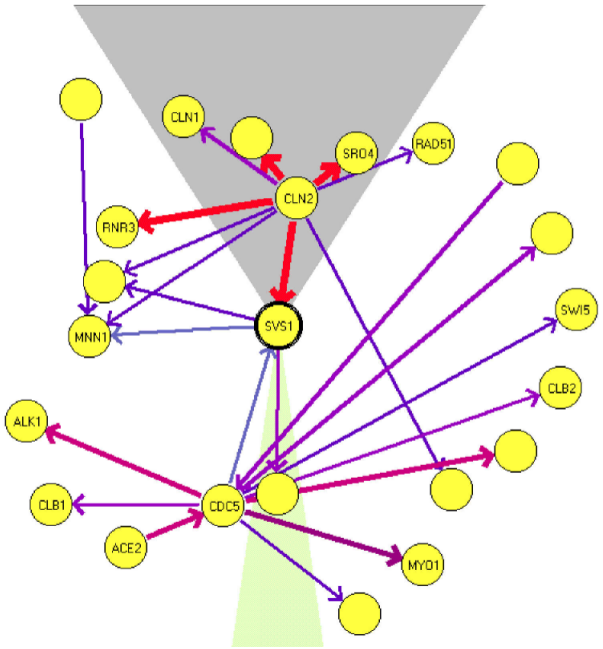

Figura 8.2.: Red Bayesiana local al gen SVS1

Puede apreciarse que el modelo permite visualizar de forma sencilla la relación entre genes y medir el grado de la misma.

8.2. *Using Bayesian networks to discover relations between genes, environment, and disease*

En este artículo se estudia las relaciones ya conocidas entre el cáncer de vejiga y la genética, además de añadir el factor medio ambiente. Para ello, se hace primero un estudio teórico previo de las redes Bayesianas y del aprendizaje. Para culminar con la aplicación a un caso de una base de datos real.

En este caso, el estudio de los genes se va a hacer mediante los polimorfismos de nucleótido único (SNPs). Los SNPs son cambios que producen una variación en un par de bases que afecta solo al 1 porciento de la población. Los científicos están estudiando cómo SNPs en el genoma humano se correlacionan con enfermedades, con la respuesta de los fármacos, y con otros fenotipos. Esta definición se basa en la definición de SNPs del *national human henome research institute* [Collins, 2021].

En el estudio, los autores van a intentar aprender las relaciones entre un número limitado de variables gen-ambiente-enfermedad recogidas como parte de un estudio de control de

casos basado en la población del cáncer de vejiga en New Hampshire. En esta base de datos hay gran variedad de SNPs, junto con atributos como el sexo, exposición al arsénico, edad, hábito de fumar, etc. El estudio se centrará en la variable referente al cáncer de vejiga.

Los autores centran el análisis en 11 variables que nos permiten explorar el papel de los SNPs en el gen XRCC3 en las posiciones 03, 04 y 241 y ERCC2/XPD en las posiciones 03, 09 y 312. Además, se estudia la exposición al arsénico, que está representada por el nivel de arsénico en las uñas de los pies. También incluyen los factores de riesgo conocidos: el sexo, la edad ($\leq$60 o $>$60) y el hábito de fumar.

En este proyecto se usa el lenguaje de programación *R* con el paquete *BNlearn* [Scutari, 2007]. Este paquete soporta los algoritmos de aprendizaje y la estimación de parámetros necesaria para el aprendizaje a través de los datos.

Los autores han decidido estudiar los algoritmos *Grow-Shrink*(GS), *Incremental Association Markov Blanket* (IAMB), *Hill-Climbing* (HC) y *Max-Min Hill-Climbing* (MMHC). Para los algoritmos basados en métricas, se ha usado la métrica $log(k2)$. Luego, definen una lista negra de relaciones entre variables que no se pueden dar.

Tras el entrenamiento, escogen la red resultante del algoritmo HC ya que consigue el mejor resultado en tres de las cuatro métricas que estudian ($log(k2)$, BIC, AIC, *log-likelihood*). Una vez seleccionado HC como el que obtiene el mejor resultado; vuelven a entrenar el algoritmo HC sin usar la lista negra. Obteniendo así una red Bayesiana intermedia.

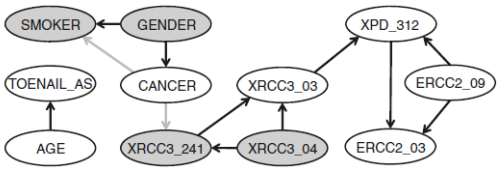

Figura 8.3.: Red Bayesiana intermedia

El resultado (Figura 8.3) muestra que tres variables (GENDER, SMOKER y XRCC3_241) tienen relación directa con CANCER, la primera como padre y las dos segundas como hijas. Además, como el padre de XRCC3_241 es XRCC3_04, esta variable está en el manto de Markov de CANCER. Por lo tanto, estas cuatro variables deben ser consideradas como posibles causas del cáncer de vejiga. Además, según un estudio anterior, los autores afirman que TOENAIL_AS (variable de exposición al arsénico) debe estar en el manto de Markov de CANCER. Por tanto, generan las redes posibles con este hecho y sin él y estudian las cuatro métricas anteriormente descritas para saber que RB es mejor. Las demás variables no mencionadas se eliminan de la RB porque no aportan información a la variable CANCER. Finalmente, obtienen una red Bayesiana final.

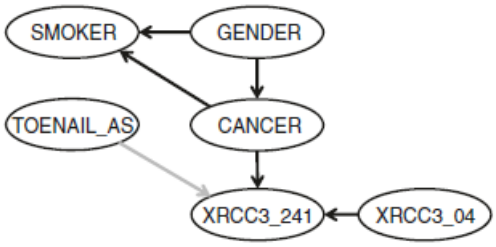

Figura 8.4.: Red Bayesiana final

En esta RB se pueden observar fácilmente las relaciones de independencia. Además, con el paquete *BNlearn* anteriormente mencionado podrían estimar los parámetros de la RB y hacer inferencia.

9. Base de datos genética

En este capítulo se va a explicar y analizar la base de datos genética de la que se aprenderá una red Bayesiana.

La base de datos de libre acceso que voy a usar se encuentra disponible en el repositorio UCI [Dua et al., 2017] y se llama *Molecular Biology (Splice-junction Gene Sequences) Data Set* [G Towell and Shavlik, 1991]. Todos los ejemplos han sido tomados del Genbank 64.1. Para entender el contenido de la base de datos, primero hay que introducir el concepto de empalme alternativo o *splicing*.

9.1. Empalme alternativo (*splicing*)

El ácido desoxirribonucleico (ADN) es la molécula que contiene la información genética de los seres vivos, imprescindible para que estos se desarrollen y realicen sus funciones. El ADN contiene la información necesaria para codificar proteínas, moléculas que participan en las diversas funciones de los organismos. Dentro del ADN se encuentran los genes, fragmentos de información que contienen las pautas necesarias para sintetizar una proteína concreta.

El ácido ribonucleico (ARN) es el que hace posible la codificación de las proteínas. Aunque el ADN contiene toda la información genética, el ARN es aquel que contiene la información que puede ser comprendida por las células. Está compuesto por una cadena simple. Para clarificar el concepto de ARN, se ha creado la siguiente figura donde se muestra el proceso de creación de la cadena simple. En este proceso, cada nucleótido pasa a ser el mismo, excepto la unión amina-timina que pasa a ser un uracilo.

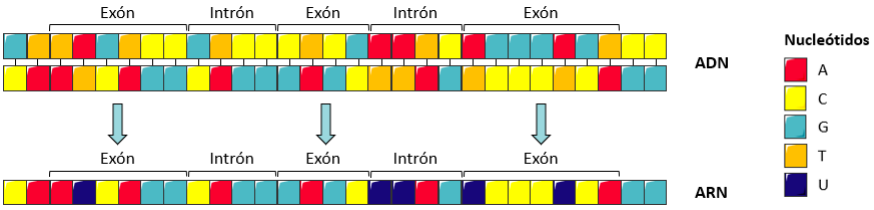

Figura 9.1.: Creación ARN

Para llegar a obtener estas proteínas son necesarios una serie de procesos que traducen la información de los genes a ARN mensajero, y posteriormente a proteínas. El denominado *splicing*, que se podría traducir como corte y empalme alternativo, es uno de los procesos que intervienen, de esta forma las regiones de ADN que codifican proteínas son seleccionadas para formar el ARN mensajero eliminando los intrones (regiones no codificantes) y uniendo los exones (regiones codificantes). Como transición entre estas regiones encontramos los *splice junctions* o regiones de empalme, es decir, secciones que cambian de un exón

a un intrón (EI) o de un intrón a un exón (IE). Para clarificar el proceso desde que se tiene el ARN, hasta que se codifica la proteína, se ha diseñado la siguiente Figura donde a partir de la cadena simple de ARN se genera la cadena de ARNm sin intrones; para, con esta cadena codificar la proteína.

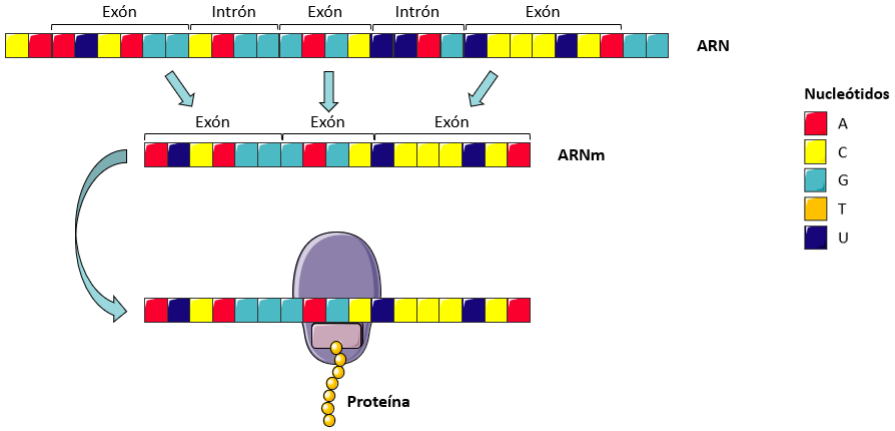

Figura 9.2.: Síntesis de proteínas

9.2. Explicación de la base de datos y análisis preliminar

La base de datos se creó para el estudio de las regiones de empalme (RE), ya vistas en el apartado anterior. Para ello, se tomaron 3190 secuencias de genes de 60 posiciones etiquetadas mediante 3 categorías. Las categorías se definen conforme al tipo de cambio que se produce en la región de empalme.

- **EI**. Denota que se ha producido en ese gen un cambio de exón a intrón.
- **IE**. Denota que se ha producido en ese gen un cambio de intrón a exón.
- **N**. En el gen no hay región de empalme y por tanto ningún cambio de exón a intrón o viceversa.

Cada una de las posiciones de la secuencia del gen toma ocho posibles valores. La tabla siguiente muestra los posibles valores y su significado.

Carácter	Significado
A	Adenina
G	Guanina
T	Timina
C	Citosina

Carácter	Significado
D	A o G o T
N	A o G o C o T
S	C o G
R	A o G

Tabla 9.1.: Posibles valores de las posiciones de la secuencia genética

Cada instancia de la base de datos se compone de tres atributos. El primero de ellos es el nombre asociado a la secuencia del gen. El segundo, es la secuencia de 60 posiciones del gen. La secuencia viene almacenada como una cadena de caracteres. El último atributo, es la etiqueta de la RE.

Nombre	Secuencia	Etiqueta
ATRINS-DONOR-521	CCAGCTGC...GCCAGTCTG	EI
ATRINS-DONOR-905	AGACCCGC...TGCCCCCGC	EI
BABAPOE-DONOR-30	GAGGTGAA...ACGGGGATG	EI
BABAPOE-DONOR-867	GGGCTGCG...GTTTTCCCC	EI
BABAPOE-DONOR-2817	GCTCAGCC...CTTGACCCT	EI
...	...	...

Tabla 9.2.: Ejemplo instancias base de datos

La base de datos que se va a estudiar es un problema de clasificación. Para ello, se tratarán cada una de las posiciones de la secuencia como variables predictoras y la variable etiqueta como variable a predecir. Dicho esto, en la figura siguiente se muestra la frecuencia de los valores que toman cada una de las posiciones para observar si hay datos extraños.

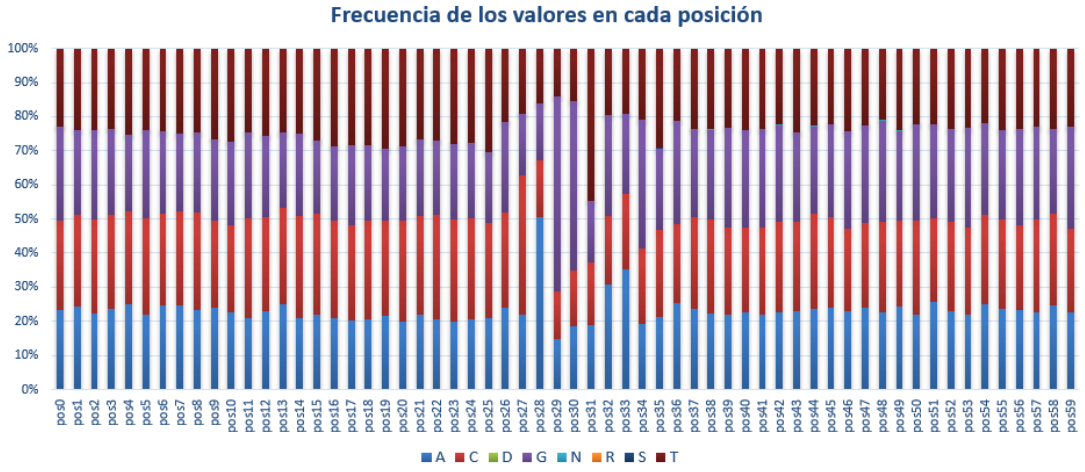

Figura 9.3.: Frecuencia de los valores por posición

En el gráfico anterior se muestran varios hechos interesantes sobre los datos. En primer lugar, los valores D, N, S y R no son relevantes en ninguna de las posiciones; ya que el porcentaje de representación en cada posición es prácticamente nulo. Será necesario comprobar si tampoco son relevantes respecto a las etiquetas. En segundo lugar, la distribución en el resto de valores para cada posición parece ser la misma; excepto, en las posiciones 27, 28, 29, 30 y 31, donde se dan algunos valores más que otros. Pero no se aprecian discordancias relevantes.

A continuación, se va a analizar el número de instancias etiquetadas con una determinada categoría. Se va a hacer para comprobar si las categorías están balanceadas o si existe una

determinada categoría con más instancias que otra. Para ello, se muestra una tabla con el número de instancias por categoría y un diagrama de sectores con la frecuencia de instancias por categoría.

	Nº Genes
EI	767
IE	768
N	1655

Tabla 9.3.: Nº de genes por etiqueta

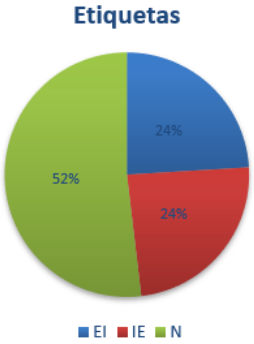

Figura 9.4.: Frecuencias etiquetas

Se puede apreciar como existe desbalanceo entre las etiquetas de las instancias. La etiqueta N, representa más de la mitad de las instancias de la base de datos. Mientras que las etiquetas IE y EI, se encuentran balanceadas entre sí. La consecuencia de este problema se va a ver reflejado en los conjuntos de entrenamiento y test. Ya que para su creación, se van a mantener las proporciones. Además, este dato habrá que ternerlo en cuenta para evaluar la bondad del modelo.

Por último, para confirmar que las instancias que toman en alguna de sus posiciones los valores D, N, S y R son irrelevantes, se presenta una tabla y un diagrama de barras con las frecuencias de aparición de uno de estos valores en una determinada categoría.

	EI (%)	IE (%)	N (%)
A	24,984	22,1534	20,577
G	25,653	31,415	22,383
T	24,273	21,771	26,445
C	25,077	24,561	30,588

	EI (%)	IE (%)	N (%)
D	0,001	0	0,002
N	0,01	0,01	0
S	0	0	0,002
R	0	0	0,002

Tabla 9.4.: Frecuencia valores posiciones por etiqueta

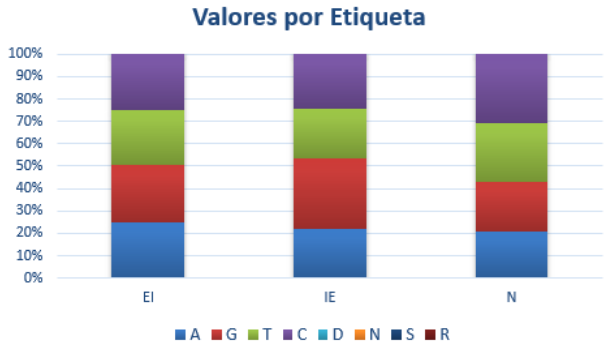

Figura 9.5.: Diagrama de barras Valores posiciones por Etiqueta

En primer lugar; en la tabla 9.4 se aprecia que los valores A, G, T y C, se distribuyen equitativamente entre las categorías. Aunque, hay algunas pequeñas variaciones al alza para las combinaciones G-IE y C-N. Además, la sospecha previamente planteada se confirma. Los valores D, N, S y R también son irrelevantes para discernir sobre las categorías debido a la baja representación que tienen en las instancias. En consecuencia, para facilitar el estudio, se decide eliminar las instancias que tienen los valores mencionados en alguna de sus posiciones. En la figura 9.5, se aprecia la misma información que en la tabla pero de forma gráfica.

10. Estudio experimental en una base de datos genética

En este capítulo se va a desarrollar el aprendizaje de la RB sobre el conjunto de datos explicado en el capítulo 9.

10.1. Introducción

En este capítulo se va a desarrollar el estudio experimental sobre los datos *Molecular Biology (Splice-junction Gene Sequences) Data Set* [G Towell and Shavlik, 1991]. Este estudio va a consistir en un preprocesamiento de los datos aplicando algunas técnicas estadísticas; para luego estimar la estructura y los parámetros de la distribución asociada a los datos. Y finalmente, valorar la bondad del modelo frente al problema de clasificación comentado en el capítulo 9. Para la experimentación, se ha desarrollado un cuaderno *Python* alojado en el siguiente enlace `https://colab.research.google.com/drive/1T_G5kHONtDNkaut8fD2qebQUQJzyP_mE?usp=sharing`, cuya arquitectura se explica a continuación.

10.2. Arquitectura del cuaderno

Un cuaderno *Python*, es un fichero donde se puede ejecutar tanto texto en formato *Markdown*, como código en lenguaje *Python*; cada conjunto de código se escribe en una celda y puede ser ejecutado por separado. En el cuaderno *Python* que se va a usar, se va a embeber un sistema para la ejecución de *R*. Esta decisión se ha tomado porque la librería *BNlearn* que se va a usar para el aprendizaje de la RB no está optimizada en *Python* y no tiene incluidas todas las funcionalidades que incluye en *R*. Además, se querían poder usar librerías de *Python* como *Pandas* y *Scikit-learn* para el preprocesamiento de los datos. Por último, para facilitar la integración de todos los paquetes y herramientas de cara a poder ser ejecutado por otro usuario fácilmente; se integró toda esta arquitectura en la nube de *google* mediante los cuadernos de *Python Google Colaboratory*. La estructura final queda de la siguiente forma.

Figura 10.1.: Arquitectura del cuaderno

En primer lugar, se destaca de la Figura 10.1 que el primer paso consiste en explicar la base de datos, que se encuentra en el capítulo 9. Luego, se realiza la carga de datos junto

con una limpieza de los mismos y una división de conjuntos de los datos; todo ello en *Python*. Finalmente, en el sistema embebido que ejecuta el lenguaje *R*; se usa la librería *BNlearn* para aprender la estructura en base a distintos algoritmos. A continuación, se elige la mejor estructura, y se estiman los parámetros de la distribución.

Una vez se ha aprendido la distribución y la estructura, ya se tiene la RB. Y por tanto, se pueden hacer peticiones a la red. En este caso se estudiará la bondad frente al problema de clasificación planteado.

A continuación, se explica el sistema por el cual se consigue ejecutar *R* y todas las librerías que se han usado en el cuaderno; junto con las funciones y parámetros que se usan de ellas.

10.2.1. Sistema *R* embebido en *Python*

La ejecución de sentencias del lenguaje *R* en *Python* se consigue con la librería *rpy2* que se encuentra en el siguiente enlace `https://rpy2.github.io/`. Al cargar la librería en *Python*, lo que se consigue es tener una interfaz entre ambos lenguajes, pudiendo así aprovechar las librerías de ambos. Cuando se habla de sistema embebido, se hace referencia a que las sentencias de *R* se están ejecutando realmente en un proceso *Python* sobre una capa de *R*.

La ejecución de *R* en el cuaderno es muy simple; ya que una vez cargado el paquete, solo es necesario añadir al principio de la celda la sentencia "%%*R*" y a continuación, el código *R* a ejecutar. Por último, la transferencia de datos a través de la interfaz es también sencilla; ya que solo hay que indicar que variable se desea transferir y en que variable se desea guardar; acompañado de un *flag* que indica si es de *R* a *Python* o viceversa.

10.3. Librerías usadas

10.3.1. Librerías *Python*

Las librerías que se van a usar en *Python* para el preprocesamiento de los datos y la evaluación de la bondad del modelo son *Pandas*, *Numpy*, *Matplotlib* y *Scikit-Learn*.

10.3.1.1. *Pandas*

La librería *Pandas* se puede encontrar en el siguiente enlace `https://pandas.pydata.org/`. Esta librería introduce la posibilidad de trabajar con estructuras de datos más complejas como los *DataFrames* y las *Series*. Los *DataFrames* son una estructura de datos que funciona como una tabla relacional. Y las *Series* son un vector uno dimensional etiquetado. Con el uso de esta librería, es muy fácil y rápido realizar tareas como resumir la información del contenido de la tabla, rellenar datos perdidos, exportar en distintos formatos, hacer uniones o separados de tablas, insertar o eliminar columnas y filas, etc.

Los *DataFrames* o tablas tienen atributos como el nombre de las columnas (*columns*) y la dimensión (*shape*). Las funciones que se van a usar sobre las tablas son:

- ***drop***. Con esta función se consigue eliminar una columna o fila.

- **Constructores**. Se usan tres constructores: uno para crear una tabla mediante un diccionario; otro mediante una matriz de datos y un vector de nombres de columnas; y otro mediante otra tabla.
- ***unique***. La función se usa para saber de que valores se compone cada columna.
- ***astype***. Su función es cambiar el tipo de una determinada columna.
- ***value_counts***. Devuelve el número de valores distintos en una columna o de un conjunto de columnas.
- ***style.highlight_max***. Para resaltar en la tabla los valores mayores.
- ***read_csv***. Con esta función se cargan los datos en formato *csv* a una tabla.

Por último, de las *Series* se necesita la función ***sort_values*** para ordenar el vector.

10.3.1.2. *Numpy*

La librería *Numpy* se va a usar para tratar los vectores N dimensionales. Esta librería contiene una gran cantidad de funciones para operar con vectores N dimensionales de forma muy compacta y sencilla. La librería se encuentra en el enlace `https://numpy.org/`.

10.3.1.3. *Matplotlib*

La librería *Matplotlib* se encuentra en el enlace `https://matplotlib.org/`. Esta librería está diseñada para la visualización de imágenes y gráficos. En el cuaderno, se va a usar para visualizar gráficos.

10.3.1.4. *Scikit-Learn*

La librería *Scikit-Learn* es una librería diseñada para el aprendizaje automático que se encuentra en el enlace `https://scikit-learn.org/stable/`. La librería incorpora funciones para el preprocesado de datos, para el entrenamiento de modelos y para el estudio de la bondad de los modelos. En el cuaderno no se van a usar funciones para el entrenamiento de los modelos.

Las clases y funciones que se van a usar para la preparación de los datos van a ser las que se describen a continuación.

- ***LabelEncoder***. Es una clase que se usa para codificar las etiquetas categóricas en enteros desde 0 al número de etiquetas -1.
- ***chi2***. Esta función permite realizar el test de independencia χ^2 sobre un conjunto de variables X y una variable y.
- ***SelectKBest***. Esta clase permite elegir las k mejores variables según un test estadístico elegido.
- ***VarianceThreshold***. Es una clase que elimina aquellas variables que tengan menor varianza que un umbral.

- ***train_test_split***. Es una función que realiza la división en dos conjuntos. Además, realiza barajados de los conjuntos y divisiones conservando las proporciones de una determinada variable.

Por último, se describen las funciones usadas para el estudio de la bondad del modelo.

- ***confusion_matrix***. Función que calcula la matriz de confusión sobre las etiquetas reales y las predichas.
- ***ConfusionMatrixDisplay***. Función para visualizar la matriz de confusión como un mapa de calor.
- ***f1_score***. Función para evaluar las etiquetas reales y predichas con la métrica *F1 Score*.
- ***recall_score***. Función para evaluar las etiquetas reales y predichas con la métrica *Recall Score*.

10.3.2. Librerías *R*

En *R*, se van a usar las librerías *BNlearn*, *graphviz*, *graph* y *grid*. Además, se usará el *Dataframe* de *R* que es igual que el de *Python* pero en este caso no es de una librería externa a *R*.

10.3.2.1. *BNlearn*

La librería *BNlearn* [Scutari, 2007], se va a usar para todo lo referente a las RBs. Es decir, para el aprendizaje de la estructura, la estimación de los parámetros de la distribución y la inferencia sobre la etiqueta a predecir. Además, será posible hacer, a través de las funciones de la librería, comparaciones entre estructuras mediante métricas o representaciones de las mismas.

Para aprender la estructura se van a usar funciones que realizan los algoritmos estudiados en el capítulo 7. Estas funciones, devuelven un objeto de tipo ***bn***; que representa una RB y se compone de los nodos, arcos, algoritmo usado y algunos atributos más que no se van a usar en este trabajo. Las funciones son **aimb**, **hc**, **mmhc** y **gs**. Además, para comparar estructuras, se usa las funciones ***compare*** y ***score*** para obtener los arcos discordantes entre dos estructuras y la evaluación de una determinada métrica sobre una estructura, respectivamente.

Para la estimación paramétrica de la distribución, se usa la función ***bn.fit***, la cual estima la distribución sobre un determinado conjunto de datos y un método. Esta función, devuelve un objeto de tipo ***bn.fit class*** con la distribución y estructura de la RB.

Para hacer inferencia se usa la función ***predict***. Los parámetros de la función son: un objeto de tipo ***bn.fit class***, una evidencia y una variable. Devuelve la probabilidad de que ocurra cada uno de los valores de esa variable.

Finalmente, las librerías *graph* y *grid* son librerías que isa internamente *BNlearn* y que serán necesarias para su utilización.

10.3.2.2. *Graphviz*

La librería *graphviz* se puede encontrar en el enlace `https://graphviz.org/`. Esta librería está diseñada para la visualización de grafos en *R* y además es usada por la librería *BNlearn* en algunas funciones para representar las estructuras de RB.

En este proyecto, se van a usar las funciones ***graphviz.compare*** y ***graphviz.plot***. Ambas funciones representan las estructuras de RBs. La primera función compara dos estructuras resaltando los arcos que pertenecen a una y otra. La segunda función visualiza una estructura de RB.

10.4. Cuaderno base de datos genética

Una vez se ha explicado la arquitectura y las librerías que se van a usar, se pasa a explicar lo realizado en el cuaderno de *Python* en *google Colaboratory* para la estimación de una RB a partir de la base de datos sobre las regiones de empalme [G Towell and Shavlik, 1991]. Para esto, se a a profundizar en cada una de las partes definidas en la arquitectura.

10.4.1. Carga de datos

Los datos se aportan por el repositorio de libre acceso UCI [Dua et al., 2017] en formato *csv*. Este formato es una forma de definir tablas estructuradas mediante fichero. Para facilitar el uso de este fichero, se ha subido a *google drive* y se ha creado un enlace compartido del mismo. El formato en el que se reciben los datos es el siguiente.

```
class ,Name, Sequence
EI ,ATRINS-DONOR-521, CCAGCTGCATCACAGGAG...CTTCGAGCCAGTC
EI ,ATRINS-DONOR-905, AGACCCGCCGGGAGGCGG...CCTCCGTGCCCCC
EI ,BABAPOE-DONOR-30, GAGGTGAAGGACGTCCTT...GGGGGCACGGGGA
EI ,BABAPOE-DONOR-867, GGGCTGCGTTGCTGGTCA...TGCTCGGTTTTCC
EI ,BABAPOE-DONOR-2817, GCTCAGCCCCCAGGTCAC...CCGGCCCTTGACC
```

Una vez se tiene claro el formato de los datos, se van a cargar en una tabla (*DataFrame*) de *pandas*. Para cargar los datos, se usa la función *read_csv* con el enlace compartido en *drive* del fichero. Obteniendo la siguiente tabla.

class	Name	Sequence
EI	ATRINS-DONOR-521	CCAGCTGCATCACAGGAG...CTTCGAGCCAGTC
EI	ATRINS-DONOR-905	AGACCCGCCGGGAGGCGG...CCTCCGTGCCCCC
EI	BABAPOE-DONOR-30	GAGGTGAAGGACGTCCTT...GGGGGCACGGGGA
EI	BABAPOE-DONOR-867	GGGCTGCGTTGCTGGTCA...TGCTCGGTTTTCC
EI	BABAPOE-DONOR-2817	GCTCAGCCCCCAGGTCAC...CCGGCCCTTGACC
...	...	...

Tabla 10.1.: Formato datos cargados

10.4.2. Limpieza y preprocesamiento de los datos

Como ya se ha expuesto en el capítulo 9, cada una de las posiciones de la secuencia es una variable predictora del problema. En la tabla 10.1, se muestra que la secuencia es una cadena de caracteres; además, la cadena contiene al inicio un espacio en blanco. En el cuaderno, la secuencia se procesa y se genera una tabla con una columna por posición. También, se va a eliminar la variable *Name*, pues no aporta información relevante. La tabla que almacena los datos a partir de estas transformaciones es la siguiente.

class	*pos*0	*pos*1	*pos*2	*pos*3	*pos*4	*pos*5	...	*pos*53	*pos*54	*pos*55	*pos*56	*pos*57	*pos*58	*pos*59
EI	C	C	A	G	C	T	...	C	A	G	T	C	T	G
EI	A	G	A	C	C	C	...	C	C	C	C	C	G	C
EI	G	A	G	G	T	G	...	G	G	G	G	A	T	G
EI	G	G	G	C	T	G	...	T	T	T	C	C	C	C
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

Tabla 10.2.: Formato tabla de variables

Luego es necesario cambiar el tipo de dato porque las variables son categóricas. Pero esta información no está presente en los datos. Para esto, se cambia el tipo de dato de las columnas a *category*.

A continuación, se van a eliminar ciertas instancias irrelevantes y se va a hacer selección de variables.

10.4.2.1. Eliminar instancias irrelevantes

En el capítulo 9 se expuso que las instancias cuyas posiciones tenían los valores *D, N, R y S* eran irrelevantes. Por tanto, en este apartado, se eliminan las instancias de la tabla cuyas posiciones tienen algún valor de los mencionados. Finalmente, la tabla pasa de tener 3190 instancias a 3175, es decir, se han eliminado únicamente 15 instancias a cambio de obtener mayor sencillez en el problema.

10.4.2.2. Selección de variables

En el problema propuesto, hay demasiadas variables predictoras; por lo tanto, es necesario realizar tests estadísticos para estudiar la posibilidad de eliminar variables que no aporten información. Para ello, se van a usar el test de independencia χ^2 y el test de varianzas. Para ambos tests se va a necesitar la codificación de las variables en enteros, además de la especificación de que las variables son categóricas, no numéricas.

La **codificación** se va a realizar mediante la clase *LabelEncoder* de *Scikit-Learn* ya explicada anteriormente. La codificación resultante es la siguiente.

valor	A	C	G	T
codificación	0	1	2	3

Una vez se ha realizado la codificación de la tabla de datos, se va a realizar el **test de independencia χ^2**. Se va a usar para conocer aquellas variables que son independientes de la variable clase. El test de independencia χ^2 es un test estadístico cuyas hipótesis son:

- H_0 : Las dos variables estudiadas por el test son independientes.
- H_1 : Las variables son dependientes.

El valor que se va a tomar para rechazar la hipótesis nula (H_0) será 0.05 (p-valor < 0.05) . El test se va a estudiar entre cada una de las variables posiciones y la variable clase. Ya que, si una variable es independiente de la clase, para el problema de clasificación sobre la clase no es necesaria esa variable.

Para implementar esta idea en el cuaderno. Primero se ha realizado el test mediante la función *chi2* de *Scikit-Learn*. Del resultado de esta función se obtienen los p-valores de cada una de las variables predictoras. A continuación, se muestra una gráfica con los p-valores de las variables en orden descendente donde se aprecia que existen variables independientes de la variable clase.

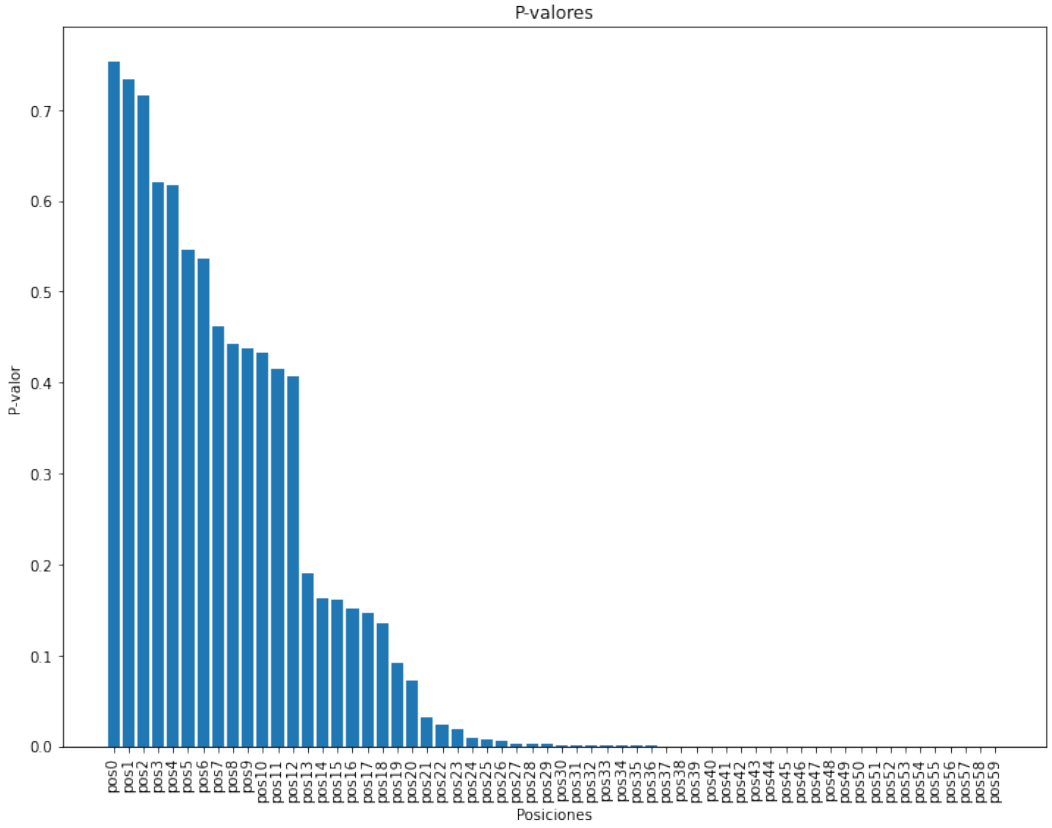

Figura 10.2.: P-valores

En segundo lugar, se cuentan aquellas variables que tienen p-valor mayor que 0.05 y que por tanto son independientes de la variable clase. Existen 21 variables cuyo p-valor es mayor que 0.05. Para eliminarlas, se usa la función *SelectKBest* de *Scikit-Learn* con el test de independencia χ^2 usando $k = 39$; porque escoge los k mejores.

Por último, se realiza el **test de varianzas** que aporta la librería *Scikit-Learn*. Se va a utilizar el test de varianzas para comprobar que cada una de las variables predictoras toman valores lo suficientemente distintos bajo un determinado umbral. El umbral que se va a tomar será del 10 %, para eliminar las variables que varíen mínimamente. El resultado de este test es que las variables cambian en al menos un 10 %. Por tanto, con el test de varianzas no se elimina ninguna variable.

10.4.3. División de conjuntos

El conjunto de datos, se va a dividir en 3 partes. Una parte para el conjunto de entramiento (*train*) con el que se estimará tanto la estructura como los parámetros de la distribución. Otro para validación (*val*) con el que se elegirá la mejor estructura. Y otro conjunto para comprobar (*test*) con el que se evaluará finalmente la bondad del modelo frente a la clasificación de las etiquetas. Para facilitar la comprensión de la división del conjunto se aporta la siguiente figura.

Figura 10.3.: División de conjuntos

Como se ve en la figura, la división será 60/20/20 respectivamente. Se ha elegido esta división porque hay suficientes datos para el entrenamiento. La creación del conjunto de validación se ha desarrollado para que los datos usados en el entrenamiento no interfieran en la elección de la estructura aprendida. Además, con el conjunto de comprobación se certifica la bondad de la RB generada ante un conjunto de datos que nunca ha estado presente en el proceso.

Esta idea se desarrolla en el cuaderno mediante la función *train_test_split* de *Scikit-Learn*. En esta función se especifica el conjunto de la variable a predecir, el conjunto de variables predictoras, la semilla para hacer el barajado de los datos y el parámetro *stratify* para mantener la proporción en la variable a predecir. Se va a mantener la proporción en esta variable porque como ya se vió en el capítulo 9, las clases están desbalanceadas.

Una vez realizada la división, el conjunto de entrenamiento queda con 1905 instancias, el de validación con 635 y el de comprobación con 635. Ahora se puede comenzar con la parte de aprendizaje de la estructura y estimación de los parámetros de la distribución.

10.4.4. Aprendizaje estructural

Como ya se ha explicado en la arquitectura del cuaderno, la parte de estimación se hace en el lenguaje R. Para ello, lo primero que se va a hacer es fijar la semilla para la reproductividad de los datos; además de transferir las estructuras de datos de *Python*, al sistema embebido

R.

La siguiente tarea, antes de poder estimar la estructura, es definir que los datos son de tipo *factor*; es decir, que los datos son categóricos. Al igual que se hizo en *Python* es necesario añadir esta información a los datos; perdida en la transferencia a *R*.

En este momento los datos ya están preparados para prodecer a estimar la estructura. Se van a estudiar los algoritmos que usa el artículo *Using Bayesian networks to discover relations between genes, environment, and disease* [Su et al., 2013] ya estudiados teóricamente en el capítulo 7. Los algoritmos son los siguientes:

- *Grow-Shrink* (GS).
- *Incremental Association Markov Blanket* (IAMB).
- *Hill-Climbing* (HC).
- *Max-Min Hill-Climbing* (MMHC).

Cada uno de ellos se van a ejecutar sobre el conjunto de entrenamiento (*train*). Además, el algoritmo basado en métricas HC, se va a entrenar con la métrica *Bayesian Information Criterion* (BIC) porque se va a buscar penalizar las estructuras complejas.

En cuanto al desarrollo de los algoritmos en *R*, se van a usar las implementaciones que aporta la librería *BNlearn* sobre los algoritmos. Para ello, se usan las funciones *gs*, *iamb*, *hc* y *mmhc* que devuelven las estructuras aprendidas en un objeto *bn*.

A continuación, se muestran las estructuras aprendidas por los algoritmos de la librería *BNlearn*.

Listing 10.1: Estructura algoritmo HC

```
model:
   [pos0][pos1|pos0][pos2|pos1][pos3|pos2][pos4|pos3][pos5|pos4][pos6|pos5]
   [pos7|pos6][pos8|pos7][pos9|pos8][pos10|pos9][pos11|pos10][pos12|pos11]
   [pos13|pos12][pos14|pos13][class|pos14][pos15|class][pos16|class]
   [pos17|class][pos18|class][pos19|class][pos20|class][pos22|class]
   [pos23|class][pos24|class][pos27|class][pos28|class][pos29|class]
   [pos30|class][pos31|class][pos32|class][pos33|class][pos34|class]
   [pos21|pos20][pos25|class:pos24][pos35|pos34][pos26|pos25][pos36|pos35]
   [pos37|pos36][pos38|pos37][pos39|pos38][pos40|pos39][pos41|pos40]
   [pos42|pos41][pos43|pos42][pos44|pos43][pos45|pos44][pos46|pos45]
   [pos47|pos46][pos48|pos47][pos49|pos48][pos50|pos49][pos51|pos50]
   [pos52|pos51][pos53|pos52][pos54|pos53][pos55|pos54][pos56|pos55]
   [pos57|pos56][pos58|pos57][pos59|pos58]
  nodes:                                61
  arcs:                                 61
    undirected arcs:                    0
    directed arcs:                      61
  average markov blanket size:          2.00
  average neighbourhood size:           2.00
  average branching factor:             1.00

  learning algorithm:                   Hill-Climbing
  score:                                BIC (disc.)
```

```
  penalization coefficient:              3.776119
  tests used in the learning procedure:  5490
  optimized:                             TRUE
```

Listing 10.2: Estructura algoritmo MMHC

```
Bayesian network learned via Hybrid methods

  model:
   [class][poso][pos1|poso][pos28|class][pos29|class][pos30|class][pos31|class]
   [pos2|pos1][pos32|pos31][pos3|pos2][pos33|pos32][pos4|pos3][pos34|pos33]
   [pos5|pos4][pos35|pos34][pos6|pos5][pos36|pos35][pos7|pos6][pos37|pos36]
   [pos8|pos7][pos38|pos37][pos9|pos8][pos39|pos38][pos10|pos9][pos40|pos39]
   [pos11|pos10][pos41|pos40][pos12|pos11][pos42|pos41][pos13|pos12]
   [pos43|pos42][pos14|pos13][pos44|pos43][pos15|pos14][pos45|pos44]
   [pos16|pos15][pos46|pos45][pos17|pos16][pos47|pos46][pos18|pos17]
   [pos48|pos47][pos19|pos18][pos49|pos48][pos20|pos19][pos50|pos49]
   [pos21|pos20][pos51|pos50][pos22|pos21][pos52|pos51][pos23|pos22]
   [pos53|pos52][pos24|pos23][pos54|pos53][pos25|pos24][pos55|pos54]
   [pos26|pos25][pos56|pos55][pos27|pos26][pos57|pos56][pos58|pos57]
   [pos59|pos58]
  nodes:                                 61
  arcs:                                  59
    undirected arcs:                     0
    directed arcs:                       59
  average markov blanket size:           1.93
  average neighbourhood size:            1.93
  average branching factor:              0.97

  learning algorithm:                    Max–Min Hill–Climbing
  constraint–based method:               Max–Min Parent Children
  conditional independence test:         Mutual Information (disc.)
  score–based method:                    Hill–Climbing
  score:                                 BIC (disc.)
  alpha threshold:                       0.05
  penalization coefficient:              3.776119
  tests used in the learning procedure:  27696
  optimized:                             TRUE
```

Listing 10.3: Estructura algoritmo GS

```
Bayesian network learned via Constraint–based methods

  model:
    [partially directed graph]
  nodes:                                 61
  arcs:                                  5
    undirected arcs:                     1
    directed arcs:                       4
  average markov blanket size:           0.23
  average neighbourhood size:            0.16
  average branching factor:              0.066

  learning algorithm:                    Grow–Shrink
  conditional independence test:         Mutual Information (disc.)
  alpha threshold:                       0.05
  tests used in the learning procedure:  4217
```

Listing 10.4: Estructura algoritmo IAMB

```
Bayesian network learned via Constraint-based methods

  model:
    [partially directed graph]
  nodes:                                61
  arcs:                                 65
    undirected arcs:                    11
    directed arcs:                      54
  average markov blanket size:          3.15
  average neighbourhood size:           2.13
  average branching factor:             0.89

  learning algorithm:                   IAMB
  conditional independence test:        Mutual Information (disc.)
  alpha threshold:                      0.05
  tests used in the learning procedure: 18263
```

Se puede observar que tanto el algoritmo GS como el algoritmo IAMB devuelven grafos parcialmente dirigidos. Esto se debe a que el algoritmo devuelve el grafo representante de la clase (parcialmente dirigido); ya que, en ciertas aristas, el test de independencia no consigue conocer la dirección correcta de la dependencia con suficiente soporte. Para solucionar esto, se usa la función *cextend*. Con esta función se consigue el grafo consistente de la clase con estructura de RB (grafo dirigido acíclico). A continuación, se muestran las estructuras aprendidas usando esta función.

Listing 10.5: Estructura algoritmo GS

```
Bayesian network learned via Constraint-based methods

  model:
   [class][pos1][pos4][pos5][pos6][pos7][pos8][pos9][pos10][pos11][pos12][pos13]
   [pos14][pos15][pos16][pos17][pos18][pos19][pos20][pos21][pos22][pos23][pos24]
   [pos25][pos26][pos27][pos28][pos29][pos30][pos31][pos32][pos33][pos34][pos35]
   [pos36][pos37][pos38][pos39][pos40][pos41][pos42][pos43][pos44][pos45][pos46]
   [pos47][pos48][pos49][pos50][pos51][pos52][pos53][pos54][pos55][pos56][pos57]
   [pos58][pos59][poso|class:pos1][pos3|class][pos2|pos1:pos3]
  nodes:                                61
  arcs:                                 5
    undirected arcs:                    0
    directed arcs:                      5
  average markov blanket size:          0.23
  average neighbourhood size:           0.16
  average branching factor:             0.082

  learning algorithm:                   Grow-Shrink
  conditional independence test:        Mutual Information (disc.)
  alpha threshold:                      0.05
  tests used in the learning procedure: 4217
```

Listing 10.6: Estructura algoritmo IAMB

```
Bayesian network learned via Constraint-based methods

  model:
   [pos1][pos3][pos5][pos8][pos11][pos13][pos15][pos17][pos19][pos22][pos25]
   [pos27][pos30][pos32][pos35][pos37][pos39][pos41][pos43][pos45][pos47][pos49]
   [pos51][pos53][pos56][pos59][class|pos30][poso|pos1][pos2|pos1:pos3:pos11]
   [pos4|pos3:pos5][pos7|pos8][pos10|pos11][pos12|pos11:pos13:pos19]
   [pos14|pos13:pos15][pos16|pos15:pos17][pos18|pos17:pos19][pos21|pos22]
   [pos24|pos25][pos26|pos25:pos27][pos36|pos35:pos37][pos38|pos37:pos39]
   [pos40|pos39:pos41][pos42|pos41:pos43][pos44|pos43:pos45][pos46|pos45:pos47]
   [pos48|pos47:pos49][pos50|pos49:pos51][pos52|pos51:pos53][pos55|pos56]
   [pos58|pos59][pos6|pos5:pos7][pos9|pos8:pos10:pos11][pos20|pos19:pos21]
   [pos23|pos22:pos24][pos29|class][pos31|class:pos30:pos32][pos34|pos35:pos58]
   [pos54|pos53:pos55][pos57|pos56:pos58][pos28|class:pos29][pos33|pos32:pos34]
  nodes:                                61
  arcs:                                 65
    undirected arcs:                    0
    directed arcs:                      65
  average markov blanket size:          3.15
  average neighbourhood size:           2.13
  average branching factor:             1.07

  learning algorithm:                   IAMB
  conditional independence test:        Mutual Information (disc.)
  alpha threshold:                      0.05
  tests used in the learning procedure: 18263
```

Seguidamente, se presentan las estructuras de RB de forma visual. En la figura 10.4 se presenta la estructura del algoritmo HC. En ella, se puede apreciar una estructura de estrella alrededor de la variable clase. Este hecho es muy prometedor a la hora de predecir con esta estructura la variable clase, pues ha captado muchas relaciones de las variables predictoras con la variable que se quiere predecir. Además, se observan relaciones directas entre posiciones contiguas de la secuencia, habrá que comprobar si las demás estructuras también conservan estas relaciones.

El algoritmo MMHC se puede visualizar en la figura 10.5. Con este algoritmo no se han detectado tantas relaciones con la variable clase. Pero parece confirmar que existen relaciones entre las posiciones contiguas de la secuencia. La estructura del algoritmo IAMB, figura 10.7, confirma lo que indican las dos estructuras anteriores sobre las relaciones entre las posiciones consecutivas de la secuencia. Pero, al igual que el algoritmo MMHC, en esta estructura no hay tantas relaciones con la variable clase como la estructura del algoritmo HC.

Se puede observar que la estructura aprendida por el algoritmo GS, figura 10.6 , no consigue captar las relaciones entre los datos. Sin embargo, a priori, todas las estructuras son posibles. Por tanto, habrá que comparar todas las estructuras y decidir en el siguiente apartado con que estructura se debe construir la distribución de la RB para que sea la mejor posible entre los algoritmos estudiados.

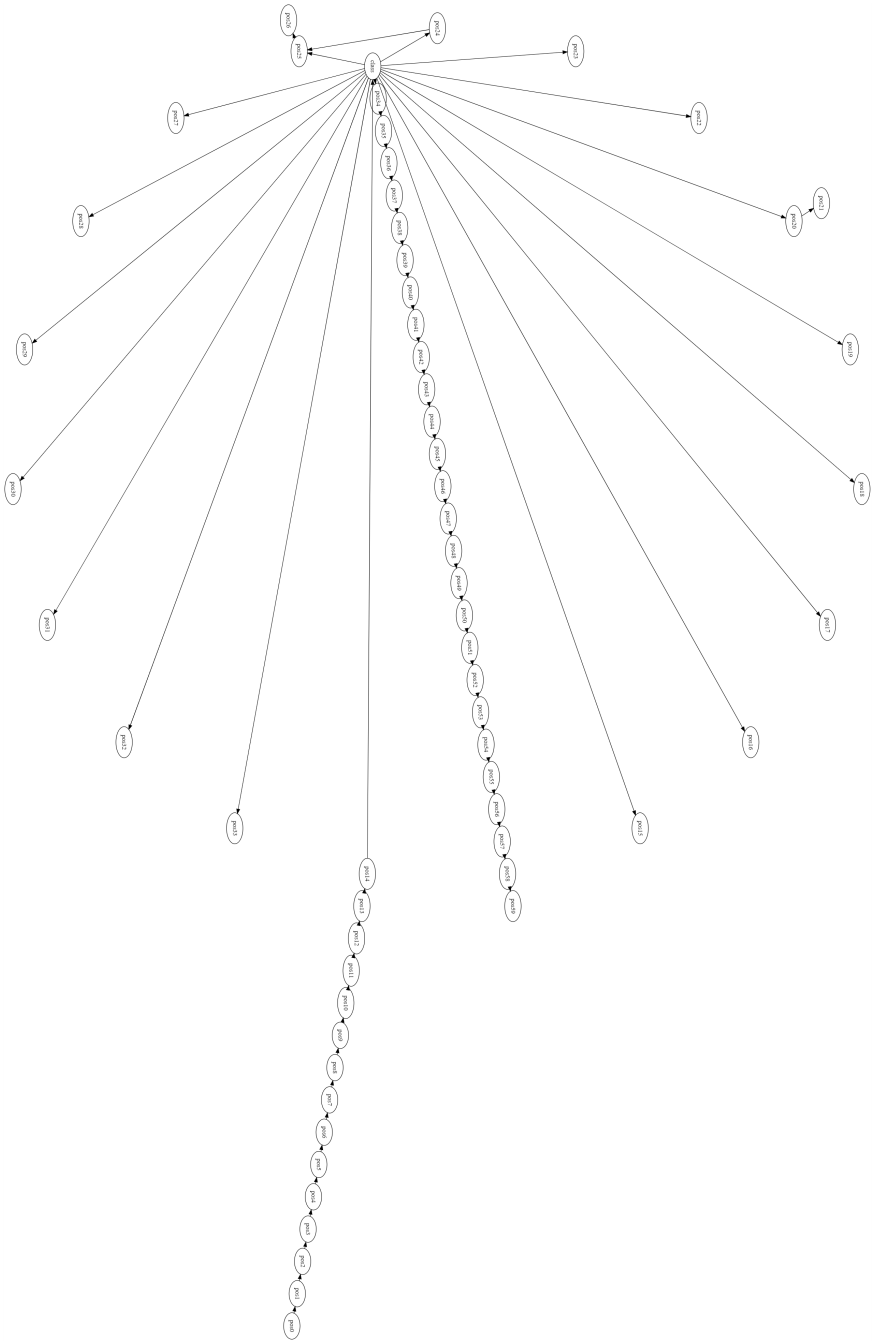

Figura 10.4.: Estructura algoritmo HC

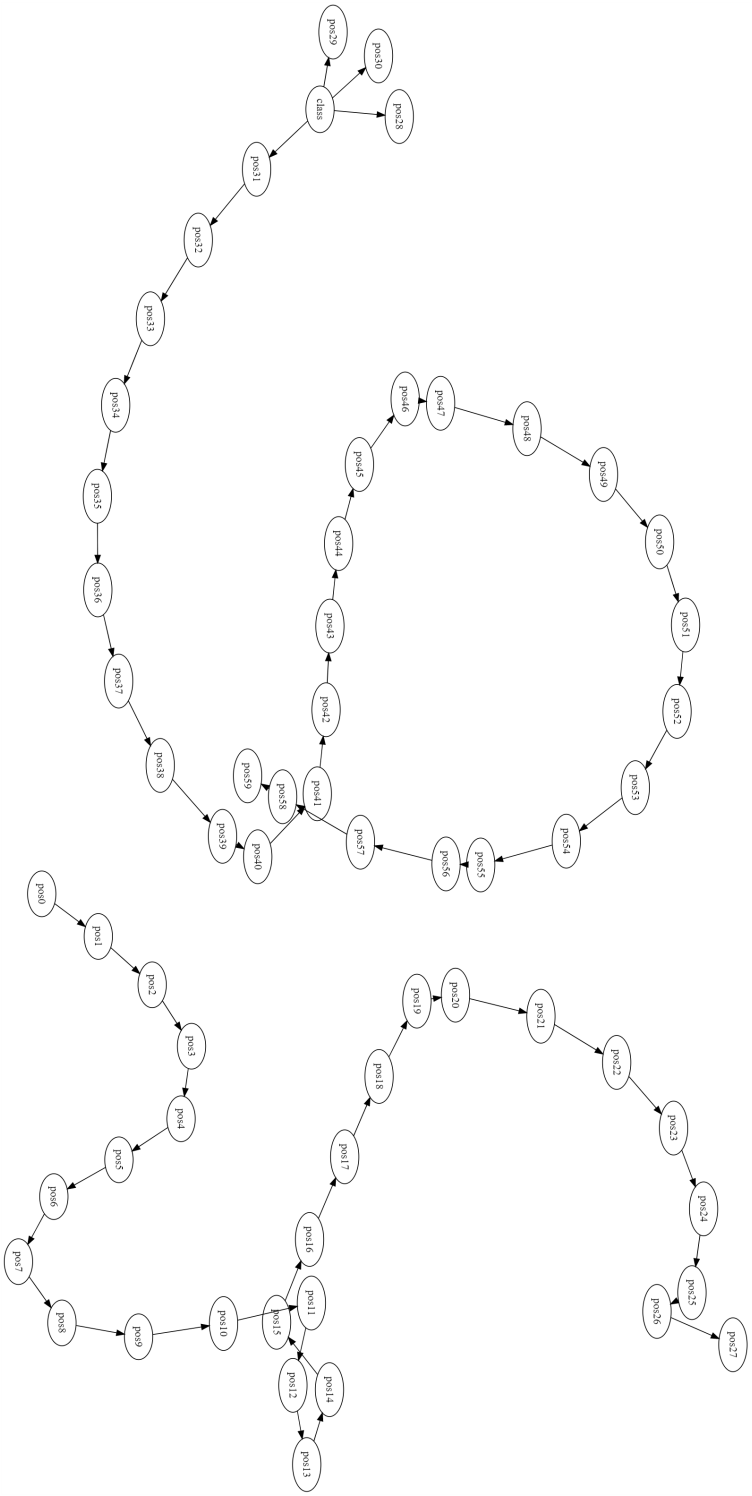

Figura 10.5.: Estructura algoritmo MMHC

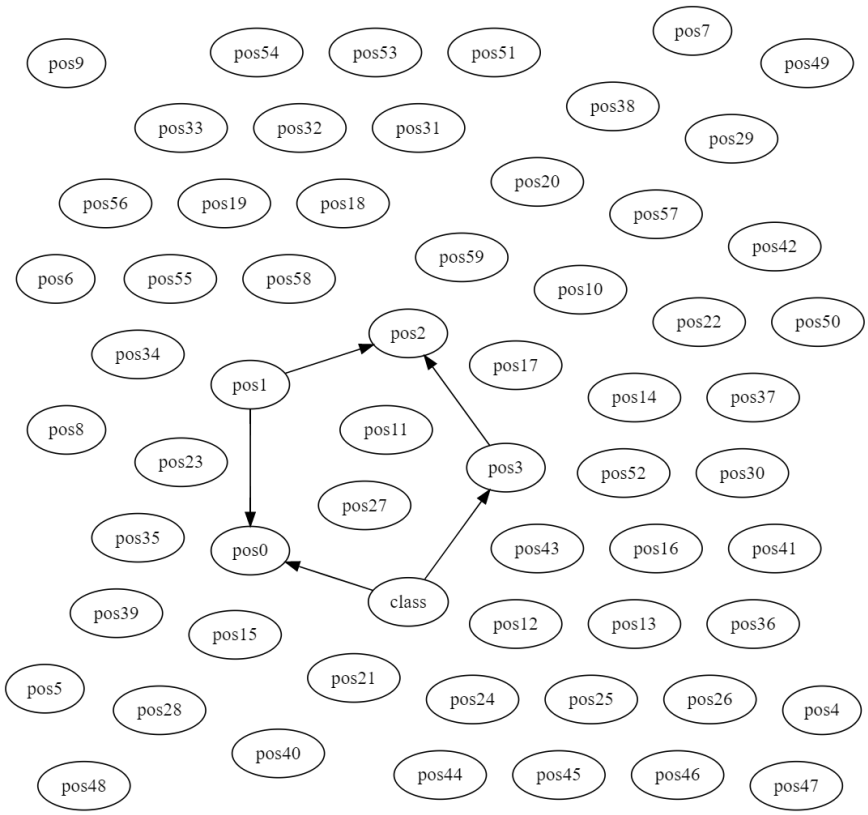

Figura 10.6.: Estructura algoritmo GS

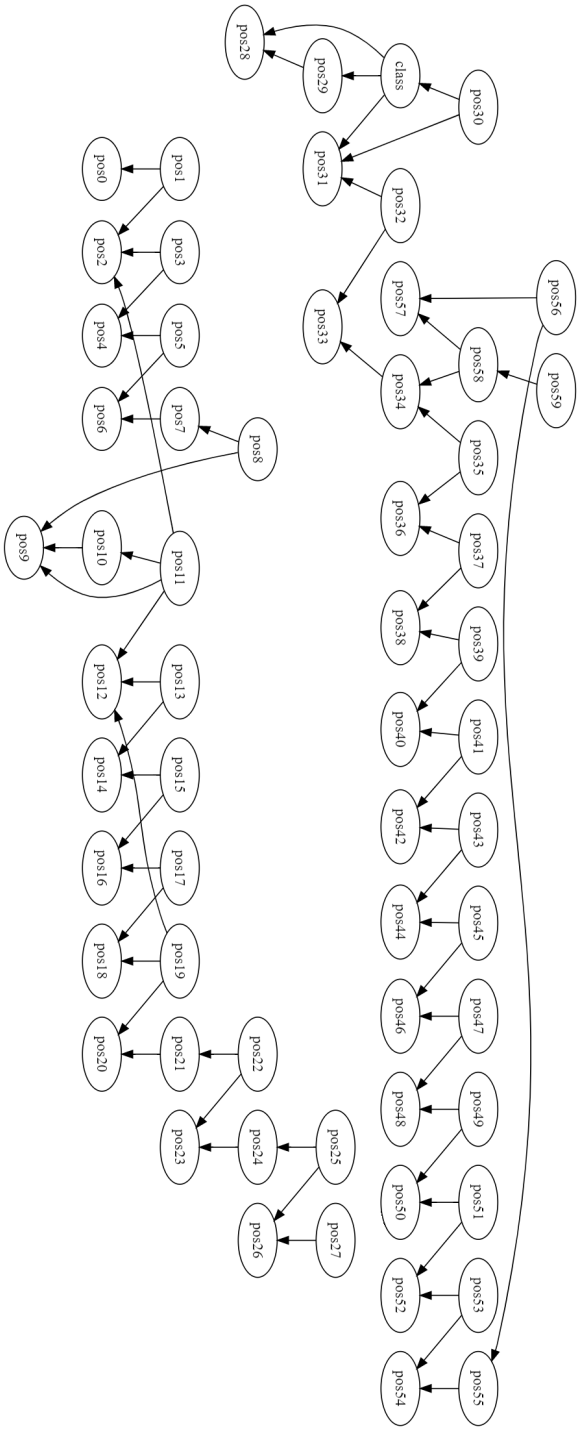

Figura 10.7.: Estructura algoritmo IAMB

10.4.4.1. Comparativa de estructuras

Para comparar las estructuras más exhaustivamente, se va a usar la función *compare* de *BNlearn*. Con ella se pueden comparar las relaciones de las estructuras teniendo como resultado el número de arcos comunes y diferentes. Además, para aquellas estructuras parecidas, usando el paquete *graphviz*, se materializan visualmente las diferencias de la siguiente manera:

- Arcos azules, son los arcos que no tiene la segunda estructura.
- Arcos rojos, son los arcos que no tiene la primera estructura.

También se ha implementado una función en *R*, para saber los arcos iguales en los que interviene la variable clase a predecir, para conocer las relaciones que apoyan la mayoría de las estructuras. A continuación, se realizan las comparaciones comentadas entre las estructuras de los algoritmos aprendidos.

- **HC con MMHC**. Ambos algoritmos aparentemente deben tener resultados parecidos porque como ya se ha estudiado, el algoritmo HC interviene en el algoritmo MMHC. Usando la función *compare*, se obtiene que las estructuras tienen 46 arcos iguales y 28 en los que difieren. Además, usando la función implementada, se aprecia que ambas estructuras tienen los siguientes arcos iguales con la variable clase.

```
     from      to
[1,] "class" "pos29"
[2,] "class" "pos28"
[3,] "class" "pos30"
[4,] "class" "pos31"
```

  Se aprecian 4 arcos con la misma relación, habrá que comprobar si más estructuras apoyan estas relaciones. Para clarificar las diferencias globales de los arcos se aporta la figura 10.8 generada con la librería *graphviz*. En esta figura se aprecia como además de coincidir en algunas relaciones con la variable clase, también coinciden con las relaciones mencionadas sobre las posiciones consecutivas de la secuencia. Aunque como se aprecia en la figura, el algoritmo mmhc capta más relaciones entre posiciones consecutivas.

- **HC con GS**. Al comparar ambas estructuras se aprecia que solo comparten un arco en general y ningún arco con la variable clase por lo que no hay nada que comparar entre ambas. Este hecho no es sorprendente; ya que, la estructura GS solo ha captado 5 relaciones en los datos.

- **HC con IAMB**. En este caso, los algoritmos han detectado 21 relaciones iguales y 84 diferentes; por lo que las estructuras generadas son bastante diferentes. Sin embargo, cabe destacar que obtienen 3 relaciones iguales con la variable clase.

```
     from      to
[1,] "class" "pos28"
[2,] "class" "pos29"
[3,] "class" "pos31"
```

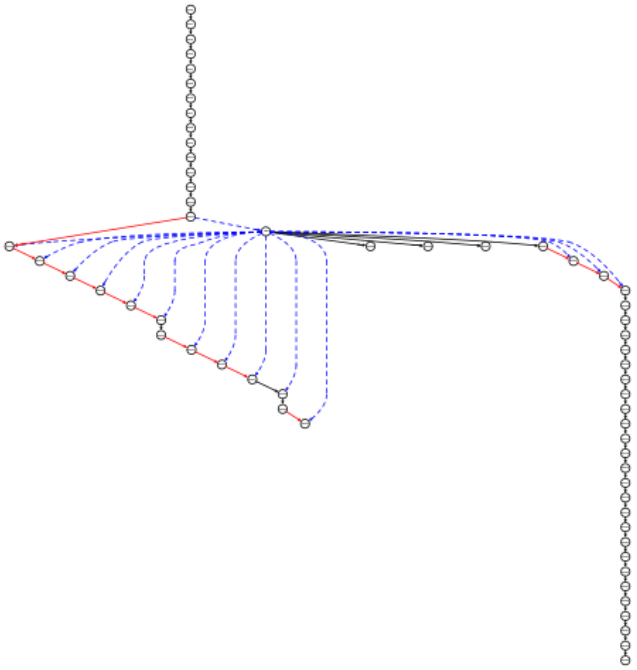

Figura 10.8.: Comparativa HC con MMHC

En la figura 10.9 se muestran las diferencias a nivel global de las estructuras generadas por los algoritmos HC y IAMB, donde se aprecia que son bastante diferentes. Además, aunque ambas deducen relaciones entre posiciones consecutivas, en la figura se aprecia que no coinciden en las direcciones. Esto es interesante pues no solo es importante la existencia de relación, también su orientación.

- **IAMB con GS**. Al igual que en el caso anterior con el algoritmo HC, al tener tan pocas relaciones el algoritmo GS, las diferencias con el algoritmo IAMB son prácticamente de todos los arcos. Además, no coinciden en ninguna relación con la variable clase. Por lo que la comparación carece de interés.

- **IAMB con MMHC**. Se tiene que ambas estructuras coinciden en 26 arcos y difieren en 72. Además, coinciden, como era de preveer, en 3 arcos con la variable clase. Los siguientes arcos se repiten en los algoritmos IAMB, HC y MMHC. Como las tres estructuras dan soporte a esta relación y además con la misma orientación. Se puede inferir que la relación entre la variable clase y las tres posiciones es fuerte.

```
     from      to
[1,] "class" "pos28"
[2,] "class" "pos29"
[3,] "class" "pos31"
```

En la figura 10.10 se observan las diferencias globales de las estructuras generadas por los algoritmos IAMB y MMHC, donde se aprecia visualmente que son muy diferentes. Al igual que con el algoritmo HC, cabe destacar que ambas coinciden con

que las relaciones entre ciertas posiciones consecutivas existen, pero no coinciden en la orientación.

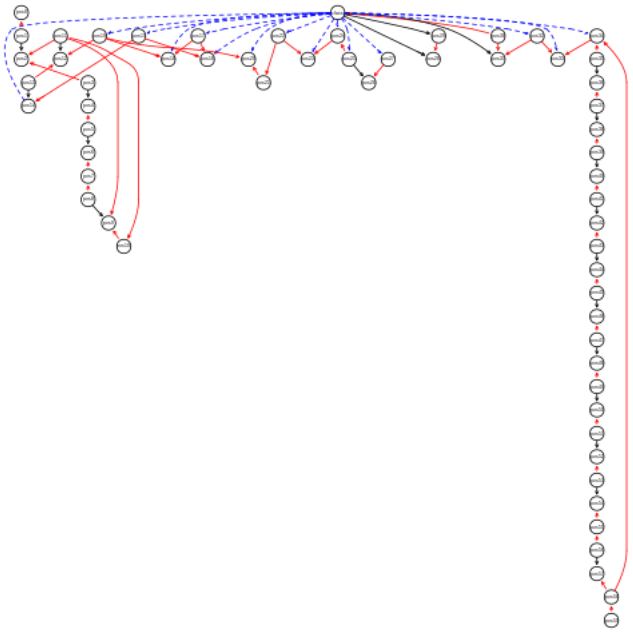

Figura 10.9.: Comparativa HC con IAMB

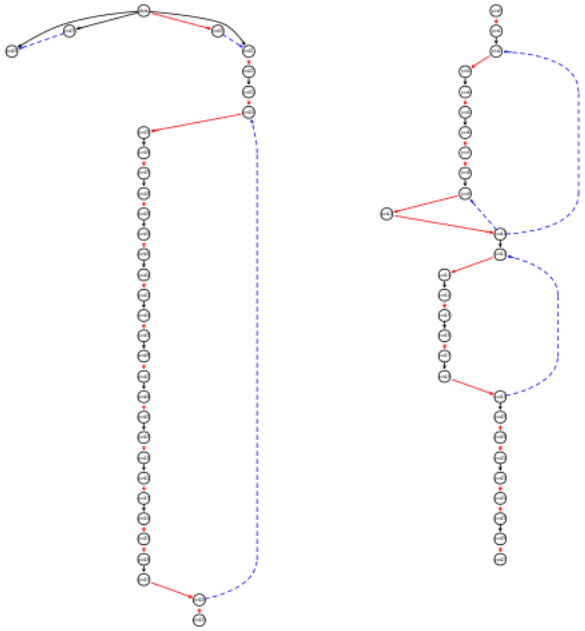

Figura 10.10.: Comparativa IAMB con MMHC

Como conclusión de la comparación de las estructuras, se puede decir que el algoritmo GS no va a ser el mejor, porque la estructura generada por él es la que menos relaciones capta entre los datos. En cuanto a los otros algoritmos, todos han captado un número considerable de relaciones; pero el que más ha captado ha sido el algoritmo HC. Por ello, paracería coherente que en el siguiente apartado fuese el seleccionado. Por último, estos algoritmos también captan las mismas relaciones entre ciertas posiciones consecutivas. Por lo que se puede afirmar que estas relaciones entre los datos existen, aunque no concuerden en la orientación de las relaciones.

10.4.4.2. Elección de la estructura de RB

Para la elección de la estructura, a simple vista, no se puede elegir una estructura en vez de otra. Por esto se definen las métricas, para poder comparar estructuras en base a un criterio común. Las métricas que se van a estudiar para decidir que estructura es la mejor son las analizadas en el capítulo 7.

- *log-likelihood (loglik)*.
- *Akaike Information Criterion* (AIC).
- *log K2* (*K2*).
- *Bayesian Information Criterion* (BIC).

Para la evaluación de las métricas se va a usar el conjunto de validación porque no ha interferido en el proceso de entrenamiento. Para la implementación en el cuaderno, se usa la función *score* de *BNlearn* junto con la métrica y el conjunto de validación. Cuanto mayor es el valor de la métrica, mejor es el algoritmo. El resultado es el siguiente.

Algoritmos	loglik	aic	k2	bic
HC	-50237.382946	-50929.382946	-51655.753168	-52850.457047
MMHC	-50527.272868	-51228.272868	-51956.548474	-53174.332038
IAMB	-49739.979075	-51682.979075	-52244.343777	-56009.675761
GS	-52638.233160	-52904.233160	-53233.494716	-53496.565285

Tabla 10.3.: Métricas sobre las estructuras de los algoritmos

Se aprecia en la tabla anterior que el algoritmo HC consigue el valor más alto en tres de las cuatro métricas (AIC, K2, BIC). Por este motivo se elige la estructura aprendida por el algoritmo HC como la que representa mejor a los datos entre las estructuras aprendidas.

10.4.5. Estimación paramétrica

Para la estimación parámetrica se va a usar el estadístico máximo verosímil (EMV) porque es el que se ha estudiado teóricamente en el capítulo 7.

Para implementar la estimación paramétrica, se va a usar la función de *BNlearn*, *bn.fit*. La cual estima los parámetros de la distribución de una determinada estructura, usando un determinado método. Como ya se ha comentado, el método sera el EMV ("*mle*") y la estructura será la aprendida con el algoritmo HC. Para estimar los parámetros se usará el conjunto de

entrenamiento.

BNlearn representa las distribuciones de probabilidad condicionada (DPCs) de la distribución en formato tablas. La distribución aprendida se encuentra en el apéndice A.

10.4.6. Inferencia y predicción

Ahora se puede realizar la predicción de la variable clase para las instancias del conjunto de comprobación (*test*). Este conjunto no ha intervenido en el proceso de estimación y por tanto es una buena forma de saber si la RB será capaz de generalizar la información aprendida.

Para la tarea de inferencia se usa la función *predict* de *BNlearn*. Con esta función se consigue la probabilidad de que se produzca un hecho ante un conjunto de evidencias. Por tanto, si la evidencia es la instancia a predecir sin la variable clase, se obtienen las probabilidades de que esa instancia pertenezca a una determinada clase. Con este hecho y la etiqueta asociada a la instancia se puede comprobar si la predicción es correcta.

Por ejemplo, si se usa la función *predict* sobre la evidencia X con el modelo aprendido anteriormente.

	Secuencia
X	TGACGCCCTCAAGGGCACTGTGAGTCCCTGCCCACCTGGGCCAGGCCCTGCCCCTTCTCT

Se obtiene una predicción de que la clase de la secuencia es *N* con probabilidad 0.999977, *EI* con probabilidad 0.226352 y *IE* con probabilidad 0. Por tanto, cogiendo la clase con mayor probabilidad se obtiene que la predicción para la secuencia es *N*. De esta manera, comparando con la clase real de la secuencia, se va a comprobar la bondad del modelo. En este caso, la clase real también es *N*.

10.4.6.1. Bondad del modelo

Para analizar la bondad del modelo en el problema de clasificación sobre las etiquetas se ha decidido implementar una función llamada *predecir* que evalúa el porcentaje de etiquetas acertadas por el modelo en un determinado conjunto.

La función *predecir* escoge la etiqueta predicha con mayor probabilidad y la compara con la etiqueta real de la instancia. Si esta etiqueta es la misma, entonces se ha acertado en la predicción. Los parámetros de la función predecir son:

- Modelo. La distribución con la que se quiere predecir.
- Conjunto. El conjunto a evaluar la predicción.
- Método. Método elegido para la función predict. Se va a usar el método *bayes-lw*.

Esta función se ha usado sobre el modelo aprendido y cada uno de los conjuntos de datos que se han definido. Los resultados se exponen a continuación.

Conjunto	entrenamiento (*train*)	validación (*val*)	comprobación (*test*)
Porcentaje acierto	96.0105	96.37795	96.06299

El modelo, consigue porcentajes altos en la predicción de la variable clase. Aunque, el único porcentaje con el que se puede analizar si el modelo es bueno, es el valor sobre el conjunto de comprobación; ya que, nunca ha intervenido en el proceso de aprendizaje.

Sin embargo, esta métrica no es suficiente para comprobar la bondad del modelo; ya que, el conjunto está desbalanceado. Para tratar esta problemática, se va contruir la matriz de confusión con las funciones *confusion_matrix* y *ConfusionMatrixDisplay* de *Scikit-Learn*. La matriz de confusión obtenida es la siguiente.

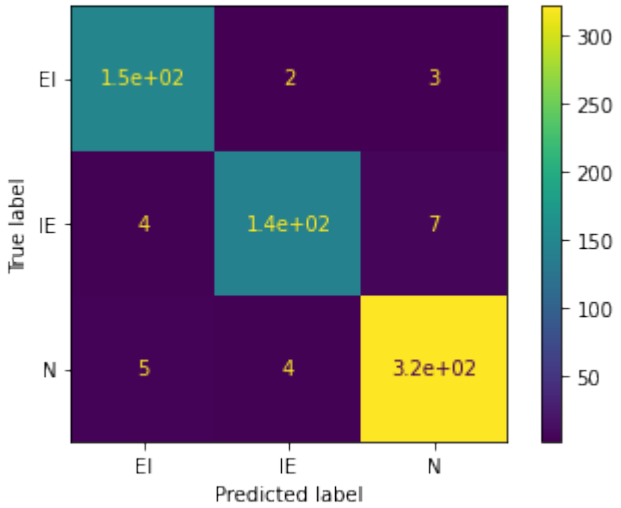

Figura 10.11.: Matriz de confusión sobre el conjunto de comprobación

El problema de la métrica anterior es que es susceptible a que un modelo consiga mas aciertos en la clase desbalanceada. Es decir, si el modelo fuese el modelo que constantemente devuelve la clase *N*; este modelo en este conjunto, tendría un 50 % de acierto. Con la matriz de confusión calculada se evita este problema; ya que, se ven los fallos que se obtienen en cada clase. Se observa, que las clases tienen prácticamente el mismo número de fallos, por lo que el modelo no se ha visto afectado por los datos desbalanceados.

Finalmente, se van a calcular dos métricas diseñadas para datos desbalanceados. Las métricas son *F1-score* y *Recall*, ambas se han usado de la librería *Scikit-Learn* especificando que los datos están desbalanceados con el parámetro *weighted*. En ambas métricas la bondad se evalúa mediante la cercanía al 1.

Métrica	***F1-score***	***Recall score***
Conjunto validación (val)	0.96057336	0.96062992

En conclusión, después de todas las pruebas realizadas, se puede afirmar que el modelo aprendido es bueno para el problema de clasificación de una secuencia de ADN de 60 posiciones sobre el tipo de empalme que contiene (ninguno, exón-intrón o intrón-exón).

11. Conclusiones y estudios futuros

Tras tener una visión completa y teórica tanto de las redes Bayesianas como del proceso de aprendizaje e inferencia sobre las mismas se confirma la utilidad del modelo matemático sobre los datos de tipo genético. Los resultados obtenidos para la base de datos *Molecular Biology (Splice-junction Gene Sequences) Data Set* [G Towell and Shavlik, 1991] son muy buenos y se ha conseguido representar las relaciones existentes entre las posiciones de la secuencia de ADN y el tipo de empalme. Además, las librerías presentan una gran cantidad de funciones para hacer el proceso de aprendizaje e inferencia sobre los datos más sencillo.

En un futuro, sería de interés analizar mas tipologías de redes Bayesianas como son las redes Bayesianas dinámicas, con semilla o temporales. Además, en el campo de la genética parece lógico seguir indagando sobre nuevas relaciones entre los datos gracias a que la red Bayesiana es un modelo de caja blanca. Por último, parece interesante comprobar o analizar desde el punto de vista de un experto en la materia las relaciones que se obtengan del aprendizaje de una red Bayesiana.

A. Primer apéndice: Distribución de probabilidad

A continuación, se muestra la distribución de probabilidad aprendida para la RB sobre los datos *Molecular Biology (Splice-junction Gene Sequences) Data Set* [G Towell and Shavlik, 1991] en el capítulo 10.

Listing A.1: DPCs de la distribución en formato tabla

```
Bayesian network parameters

  Parameters of node class (multinomial distribution)

Conditional probability table:

     pos14
class         A         C         G         T
   EI 0.2549020 0.2539130 0.2729258 0.1788793
   IE 0.1519608 0.3286957 0.1419214 0.3081897
   N  0.5931373 0.4173913 0.5851528 0.5129310

  Parameters of node poso (multinomial distribution)

Conditional probability table:
         A         C         G         T
0.2293963 0.2745407 0.2656168 0.2304462

  Parameters of node pos1 (multinomial distribution)

Conditional probability table:

    poso
pos1          A         C         G         T
   A 0.26773455 0.23518164 0.31027668 0.14350797
   C 0.21281465 0.35564054 0.21146245 0.28701595
   G 0.29061785 0.09751434 0.31027668 0.32801822
   T 0.22883295 0.31166348 0.16798419 0.24145786

  Parameters of node pos2 (multinomial distribution)

Conditional probability table:

    pos1
pos2          A         C         G         T
   A 0.2782609 0.2519531 0.2338205 0.1387665
   C 0.2217391 0.3203125 0.2964509 0.2709251
   G 0.3130435 0.1250000 0.3110647 0.3414097
   T 0.1869565 0.3027344 0.1586639 0.2488987

  Parameters of node pos3 (multinomial distribution)

Conditional probability table:

    pos2
pos3          A         C         G         T
```

```
   A 0.2384259 0.2749529 0.2070312 0.1627907
   C 0.2384259 0.3182674 0.2773438 0.2767442
   G 0.2986111 0.1111111 0.3457031 0.3023256
   T 0.2245370 0.2956685 0.1699219 0.2581395

  Parameters of node pos4 (multinomial distribution)

Conditional probability table:

    pos3
pos4         A         C         G         T
   A 0.2752941 0.2589118 0.3050505 0.1769912
   C 0.2282353 0.3151970 0.2707071 0.3119469
   G 0.3058824 0.1013133 0.2323232 0.2566372
   T 0.1905882 0.3245779 0.1919192 0.2544248

  Parameters of node pos5 (multinomial distribution)

Conditional probability table:

    pos4
pos5         A         C         G         T
   A 0.2674897 0.2351852 0.2819277 0.1357759
   C 0.2366255 0.3370370 0.2843373 0.2737069
   G 0.2880658 0.1018519 0.2626506 0.3340517
   T 0.2078189 0.3259259 0.1710843 0.2564655

  Parameters of node pos6 (multinomial distribution)

Conditional probability table:

    pos5
pos6         A         C         G         T
   A 0.2883295 0.2785978 0.2483660 0.1563169
   C 0.2059497 0.2933579 0.2984749 0.2483940
   G 0.2860412 0.1143911 0.2832244 0.3319058
   T 0.2196796 0.3136531 0.1699346 0.2633833

  Parameters of node pos7 (multinomial distribution)

Conditional probability table:

    pos6
pos7         A         C         G         T
   A 0.2974138 0.2509960 0.2605932 0.1477516
   C 0.2521552 0.3007968 0.2944915 0.2805139
   G 0.2543103 0.1115538 0.2690678 0.3147752
   T 0.1961207 0.3366534 0.1758475 0.2569593

  Parameters of node pos8 (multinomial distribution)

Conditional probability table:

    pos7
pos8          A          C          G          T
   A 0.25657895 0.28252788 0.23660714 0.12095032
   C 0.25438596 0.33271375 0.24107143 0.28077754
   G 0.26973684 0.08364312 0.31250000 0.30885529
   T 0.21929825 0.30111524 0.20982143 0.28941685
```

```
  Parameters of node pos9 (multinomial distribution)

Conditional probability table:

    pos8
pos9         A         C         G         T
   A 0.2691415 0.2795497 0.2660754 0.1448980
   C 0.2691415 0.2739212 0.2461197 0.2857143
   G 0.2575406 0.1219512 0.2993348 0.2857143
   T 0.2041763 0.3245779 0.1884701 0.2836735

  Parameters of node pos10 (multinomial distribution)

Conditional probability table:

     pos9
pos10         A         C         G         T
    A 0.2741228 0.2612086 0.2572062 0.1298969
    C 0.2214912 0.3157895 0.2439024 0.2742268
    G 0.2741228 0.1052632 0.2638581 0.3216495
    T 0.2302632 0.3177388 0.2350333 0.2742268

  Parameters of node pos11 (multinomial distribution)

Conditional probability table:

     pos10
pos11         A         C         G         T
    A 0.23287671 0.24703557 0.24889868 0.11242604
    C 0.24657534 0.32213439 0.29515419 0.28205128
    G 0.30821918 0.09683794 0.28854626 0.34714004
    T 0.21232877 0.33399209 0.16740088 0.25838264

  Parameters of node pos12 (multinomial distribution)

Conditional probability table:

     pos11
pos12         A         C         G         T
    A 0.28967254 0.24452555 0.27087576 0.15138593
    C 0.22670025 0.32481752 0.25661914 0.28144989
    G 0.26196474 0.08576642 0.30142566 0.29637527
    T 0.22166247 0.34489051 0.17107943 0.27078891

  Parameters of node pos13 (multinomial distribution)

Conditional probability table:

     pos12
pos13         A         C         G         T
    A 0.30684327 0.27376426 0.33561644 0.11270492
    C 0.22737307 0.31749049 0.24885845 0.32991803
    G 0.29139073 0.07984791 0.27168950 0.29098361
    T 0.17439294 0.32889734 0.14383562 0.26639344

  Parameters of node pos14 (multinomial distribution)

Conditional probability table:

     pos13
```

```
pos14          A          C          G          T
    A 0.23298969 0.25370370 0.24597701 0.11460674
    C 0.26185567 0.35370370 0.28735632 0.29662921
    G 0.27835052 0.09259259 0.29655172 0.32359551
    T 0.22680412 0.30000000 0.17011494 0.26516854

  Parameters of node pos15 (multinomial distribution)

Conditional probability table:

     class
pos15        EI        IE         N
    A 0.2423581 0.1111111 0.2510121
    C 0.3253275 0.3790850 0.2570850
    G 0.1921397 0.1111111 0.2621457
    T 0.2401747 0.3986928 0.2297571

  Parameters of node pos16 (multinomial distribution)

Conditional probability table:

     class
pos16         EI         IE         N
    A 0.20742358 0.09368192 0.25000000
    C 0.31441048 0.37908497 0.25101215
    G 0.28602620 0.10675381 0.22773279
    T 0.19213974 0.42047930 0.27125506

  Parameters of node pos17 (multinomial distribution)

Conditional probability table:

     class
pos17         EI         IE         N
    A 0.24017467 0.07189542 0.23380567
    C 0.27074236 0.33769063 0.25000000
    G 0.25982533 0.12636166 0.26821862
    T 0.22925764 0.46405229 0.24797571

  Parameters of node pos18 (multinomial distribution)

Conditional probability table:

     class
pos18         EI         IE         N
    A 0.25327511 0.07407407 0.23178138
    C 0.25327511 0.37037037 0.26518219
    G 0.26637555 0.10239651 0.24696356
    T 0.22707424 0.45315904 0.25607287

  Parameters of node pos19 (multinomial distribution)

Conditional probability table:

     class
pos19         EI         IE         N
    A 0.25764192 0.05010893 0.27024291
    C 0.28602620 0.36383442 0.22368421
    G 0.23144105 0.11982571 0.25202429
    T 0.22489083 0.46623094 0.25404858
```

```
  Parameters of node pos20 (multinomial distribution)

Conditional probability table:

     class
pos20         EI         IE          N
    A 0.22052402 0.09150327 0.24898785
    C 0.27292576 0.35294118 0.26720648
    G 0.30567686 0.09803922 0.25809717
    T 0.20087336 0.45751634 0.22570850

  Parameters of node pos21 (multinomial distribution)

Conditional probability table:

     pos20
pos21          A          C          G          T
    A 0.27763496 0.26497278 0.25909091 0.10476190
    C 0.22879177 0.35934664 0.28863636 0.27809524
    G 0.29562982 0.07259528 0.27272727 0.29142857
    T 0.19794344 0.30308530 0.17954545 0.32571429

  Parameters of node pos22 (multinomial distribution)

Conditional probability table:

     class
pos22         EI         IE          N
    A 0.25109170 0.04793028 0.24493927
    C 0.31877729 0.43355120 0.26113360
    G 0.22707424 0.11328976 0.25506073
    T 0.20305677 0.40522876 0.23886640

  Parameters of node pos23 (multinomial distribution)

Conditional probability table:

     class
pos23         EI         IE          N
    A 0.20960699 0.08278867 0.24898785
    C 0.26637555 0.43355120 0.23987854
    G 0.31004367 0.06753813 0.24696356
    T 0.21397380 0.41612200 0.26417004

  Parameters of node pos24 (multinomial distribution)

Conditional probability table:

     class
pos24         EI         IE          N
    A 0.28384279 0.05228758 0.23076923
    C 0.21397380 0.45751634 0.26113360
    G 0.25764192 0.06318083 0.29453441
    T 0.24454148 0.42701525 0.21356275

  Parameters of node pos25 (multinomial distribution)

Conditional probability table:
```

```
, , pos24 = A

      class
pos25          EI          IE          N
    A 0.32307692 0.20833333 0.26754386
    C 0.15384615 0.37500000 0.19298246
    G 0.26923077 0.00000000 0.28070175
    T 0.25384615 0.41666667 0.25877193

, , pos24 = C

      class
pos25          EI          IE          N
    A 0.33673469 0.08095238 0.30232558
    C 0.20408163 0.42857143 0.30620155
    G 0.06122449 0.00952381 0.09689922
    T 0.39795918 0.48095238 0.29457364

, , pos24 = G

      class
pos25          EI          IE          N
    A 0.31355932 0.10344828 0.30240550
    C 0.24576271 0.31034483 0.23711340
    G 0.33898305 0.37931034 0.26460481
    T 0.10169492 0.20689655 0.19587629

, , pos24 = T

      class
pos25          EI          IE          N
    A 0.08035714 0.03571429 0.12322275
    C 0.31250000 0.27551020 0.23696682
    G 0.33035714 0.10714286 0.36018957
    T 0.27678571 0.58163265 0.27962085


  Parameters of node pos26 (multinomial distribution)

Conditional probability table:

      pos25
pos26          A         C         G         T
    A 0.2832512 0.2933071 0.2284264 0.1423786
    C 0.2364532 0.3326772 0.2563452 0.2663317
    G 0.2955665 0.1200787 0.3477157 0.3634841
    T 0.1847291 0.2539370 0.1675127 0.2278057

  Parameters of node pos27 (multinomial distribution)

Conditional probability table:

      class
pos27          EI          IE          N
    A 0.338427948 0.032679739 0.268218623
    C 0.368995633 0.773420479 0.246963563
    G 0.179039301 0.004357298 0.257085020
    T 0.113537118 0.189542484 0.227732794

  Parameters of node pos28 (multinomial distribution)
```

```
Conditional probability table:

     class
pos28          EI          IE          N
    A 0.587336245 0.997821351 0.243927126
    C 0.120087336 0.002178649 0.265182186
    G 0.155021834 0.000000000 0.243927126
    T 0.137554585 0.000000000 0.246963563

  Parameters of node pos29 (multinomial distribution)

Conditional probability table:

     class
pos29          EI          IE          N
    A 0.082969432 0.002178649 0.247975709
    C 0.030567686 0.002178649 0.254048583
    G 0.827510917 0.995642702 0.266194332
    T 0.058951965 0.000000000 0.231781377

  Parameters of node pos30 (multinomial distribution)

Conditional probability table:

     class
pos30          EI          IE          N
    A 0.000000000 0.307189542 0.215587045
    C 0.002183406 0.145969499 0.240890688
    G 0.997816594 0.448801743 0.280364372
    T 0.000000000 0.098039216 0.263157895

  Parameters of node pos31 (multinomial distribution)

Conditional probability table:

     class
pos31          EI          IE          N
    A 0.002183406 0.254901961 0.253036437
    C 0.006550218 0.204793028 0.242914980
    G 0.002183406 0.220043573 0.245951417
    T 0.989082969 0.320261438 0.258097166

  Parameters of node pos32 (multinomial distribution)

Conditional probability table:

     class
pos32          EI          IE          N
    A 0.48908297 0.23965142 0.27226721
    C 0.02183406 0.27015251 0.24392713
    G 0.47161572 0.23965142 0.26214575
    T 0.01746725 0.25054466 0.22165992

  Parameters of node pos33 (multinomial distribution)

Conditional probability table:

     class
pos33          EI          IE          N
```

```
    A 0.72925764 0.23965142 0.24696356
    C 0.09170306 0.26361656 0.25809717
    G 0.12008734 0.26361656 0.26012146
    T 0.05895197 0.23311547 0.23481781

  Parameters of node pos34 (multinomial distribution)

Conditional probability table:

     class
pos34         EI         IE          N
    A 0.04803493 0.22222222 0.24898785
    C 0.05458515 0.30936819 0.26113360
    G 0.84716157 0.21132898 0.24190283
    T 0.05021834 0.25708061 0.24797571

  Parameters of node pos35 (multinomial distribution)

Conditional probability table:

     pos34
pos35          A          C          G          T
    A 0.24054054 0.27058824 0.20027624 0.14507772
    C 0.23243243 0.33882353 0.20165746 0.29274611
    G 0.30270270 0.08941176 0.25000000 0.33678756
    T 0.22432432 0.30117647 0.34806630 0.22538860

  Parameters of node pos36 (multinomial distribution)

Conditional probability table:

     pos35
pos36         A         C         G         T
    A 0.3086420 0.2801636 0.2950108 0.1672727
    C 0.1753086 0.2822086 0.2342733 0.2000000
    G 0.3481481 0.1615542 0.3253796 0.4272727
    T 0.1679012 0.2760736 0.1453362 0.2054545

  Parameters of node pos37 (multinomial distribution)

Conditional probability table:

     pos36
pos37          A          C          G          T
    A 0.24285714 0.29274005 0.24793388 0.13054830
    C 0.19387755 0.33021077 0.28595041 0.30809399
    G 0.31632653 0.09133489 0.30082645 0.28981723
    T 0.24693878 0.28571429 0.16528926 0.27154047

  Parameters of node pos38 (multinomial distribution)

Conditional probability table:

     pos37
pos38         A         C         G         T
    A 0.2454955 0.2220114 0.2525667 0.1454139
    C 0.2319820 0.3396584 0.2484600 0.2975391
    G 0.3018018 0.1176471 0.3182752 0.3199105
    T 0.2207207 0.3206831 0.1806982 0.2371365
```

```
  Parameters of node pos39 (multinomial distribution)

Conditional probability table:

     pos38
pos39         A         C         G         T
    A 0.2657005 0.2611940 0.2226721 0.1084599
    C 0.1884058 0.3227612 0.2469636 0.2472885
    G 0.3236715 0.1473881 0.3259109 0.3991323
    T 0.2222222 0.2686567 0.2044534 0.2451193

  Parameters of node pos40 (multinomial distribution)

Conditional probability table:

     pos39
pos40         A         C         G         T
    A 0.3097561 0.2546201 0.2222222 0.1488889
    C 0.2170732 0.3203285 0.2491039 0.1955556
    G 0.2780488 0.1047228 0.3584229 0.3955556
    T 0.1951220 0.3203285 0.1702509 0.2600000

  Parameters of node pos41 (multinomial distribution)

Conditional probability table:

     pos40
pos41         A         C         G         T
    A 0.2420814 0.2563559 0.2449355 0.1696429
    C 0.2058824 0.3389831 0.2615101 0.2254464
    G 0.3484163 0.1207627 0.3057090 0.3549107
    T 0.2036199 0.2838983 0.1878453 0.2500000

  Parameters of node pos42 (multinomial distribution)

Conditional probability table:

     pos41
pos42         A         C         G         T
    A 0.2334096 0.2631579 0.2574627 0.1484018
    C 0.2173913 0.3360324 0.2742537 0.2557078
    G 0.3501144 0.1153846 0.3190299 0.3401826
    T 0.1990847 0.2854251 0.1492537 0.2557078

  Parameters of node pos43 (multinomial distribution)

Conditional probability table:

     pos42
pos43         A         C         G         T
    A 0.2620690 0.2230769 0.2924528 0.1357143
    C 0.2091954 0.3557692 0.2320755 0.2428571
    G 0.2988506 0.1192308 0.2981132 0.3190476
    T 0.2298851 0.3019231 0.1773585 0.3023810

  Parameters of node pos44 (multinomial distribution)

Conditional probability table:

     pos43
```

```
pos44          A         C         G         T
    A 0.3099548 0.2475050 0.2417355 0.1401674
    C 0.2081448 0.3532934 0.2954545 0.2594142
    G 0.2737557 0.1137725 0.3099174 0.3661088
    T 0.2081448 0.2854291 0.1528926 0.2343096

  Parameters of node pos45 (multinomial distribution)

Conditional probability table:

     pos44
pos45          A         C         G         T
    A 0.2651685 0.2611940 0.2743539 0.1282660
    C 0.2112360 0.3544776 0.2326044 0.2684086
    G 0.3213483 0.1119403 0.3359841 0.3301663
    T 0.2022472 0.2723881 0.1570577 0.2731591

  Parameters of node pos46 (multinomial distribution)

Conditional probability table:

     pos45
pos46          A         C         G         T
    A 0.3066667 0.2431907 0.2583170 0.1418605
    C 0.1666667 0.3404669 0.2035225 0.2418605
    G 0.2866667 0.1284047 0.3737769 0.3348837
    T 0.2400000 0.2879377 0.1643836 0.2813953

  Parameters of node pos47 (multinomial distribution)

Conditional probability table:

     pos46
pos47          A         C         G         T
    A 0.2719298 0.2860262 0.2679245 0.1431670
    C 0.2061404 0.3013100 0.2716981 0.2299349
    G 0.3355263 0.1091703 0.3433962 0.3535792
    T 0.1864035 0.3034934 0.1169811 0.2733189

  Parameters of node pos48 (multinomial distribution)

Conditional probability table:

     pos47
pos48          A         C         G         T
    A 0.2419006 0.2800830 0.2299270 0.1529126
    C 0.2095032 0.3485477 0.2445255 0.2451456
    G 0.3563715 0.1244813 0.3412409 0.3446602
    T 0.1922246 0.2468880 0.1843066 0.2572816

  Parameters of node pos49 (multinomial distribution)

Conditional probability table:

     pos48
pos49          A         C         G         T
    A 0.2614679 0.2920000 0.2563177 0.1614458
    C 0.2064220 0.3220000 0.2220217 0.2048193
    G 0.3073394 0.1400000 0.3231047 0.3542169
    T 0.2247706 0.2460000 0.1985560 0.2795181
```

```
  Parameters of node pos50 (multinomial distribution)

Conditional probability table:

     pos49
pos50          A          C          G          T
    A 0.24733475 0.29629630 0.23396226 0.09843400
    C 0.25586354 0.33333333 0.26226415 0.25727069
    G 0.32835821 0.08278867 0.33773585 0.38926174
    T 0.16844350 0.28758170 0.16603774 0.25503356

  Parameters of node pos51 (multinomial distribution)

Conditional probability table:

     pos50
pos51          A          C          G          T
    A 0.2380952 0.3358634 0.2990826 0.1355932
    C 0.2047619 0.3055028 0.2348624 0.2227603
    G 0.3380952 0.1214421 0.2990826 0.3559322
    T 0.2190476 0.2371917 0.1669725 0.2857143

  Parameters of node pos52 (multinomial distribution)

Conditional probability table:

     pos51
pos52          A          C          G          T
    A 0.2641129 0.2612420 0.2596899 0.1267606
    C 0.2137097 0.3190578 0.2441860 0.2112676
    G 0.3064516 0.1199143 0.3294574 0.3849765
    T 0.2157258 0.2997859 0.1666667 0.2769953

  Parameters of node pos53 (multinomial distribution)

Conditional probability table:

     pos52
pos53          A          C          G          T
    A 0.2607710 0.2590234 0.2306273 0.1352550
    C 0.2063492 0.3375796 0.2472325 0.2638581
    G 0.3560091 0.1549894 0.3339483 0.3547672
    T 0.1768707 0.2484076 0.1881919 0.2461197

  Parameters of node pos54 (multinomial distribution)

Conditional probability table:

     pos53
pos54          A          C          G          T
    A 0.2931442 0.2644135 0.2819615 0.1544118
    C 0.2198582 0.3598410 0.2381786 0.2573529
    G 0.3238771 0.1133201 0.3064799 0.3259804
    T 0.1631206 0.2624254 0.1733800 0.2622549

  Parameters of node pos55 (multinomial distribution)

Conditional probability table:
```

```
     pos54
pos55          A         C         G         T
    A 0.3118503 0.2660194 0.2529880 0.1474201
    C 0.1850312 0.3242718 0.2330677 0.2555283
    G 0.3014553 0.1126214 0.3406375 0.3120393
    T 0.2016632 0.2970874 0.1733068 0.2850123

  Parameters of node pos56 (multinomial distribution)

Conditional probability table:

     pos55
pos56          A         C         G         T
    A 0.27637131 0.30188679 0.24351297 0.14569536
    C 0.20675105 0.29350105 0.22554890 0.24944812
    G 0.32700422 0.09224319 0.34331337 0.34878587
    T 0.18987342 0.31236897 0.18762475 0.25607064

  Parameters of node pos57 (multinomial distribution)

Conditional probability table:

     pos56
pos57          A         C         G         T
    A 0.2267819 0.2866379 0.2306238 0.1492205
    C 0.2203024 0.3168103 0.2741021 0.2717149
    G 0.3434125 0.1271552 0.3100189 0.3251670
    T 0.2095032 0.2693966 0.1852552 0.2538976

  Parameters of node pos58 (multinomial distribution)

Conditional probability table:

     pos57
pos58          A         C         G         T
    A 0.24121780 0.28682171 0.30113636 0.13824885
    C 0.23887588 0.33527132 0.22727273 0.29953917
    G 0.31615925 0.09496124 0.30871212 0.27649770
    T 0.20374707 0.28294574 0.16287879 0.28571429

  Parameters of node pos59 (multinomial distribution)

Conditional probability table:

     pos58
pos59          A         C         G         T
    A 0.2617021 0.2628571 0.2162741 0.1106095
    C 0.2148936 0.2933333 0.2355460 0.2415350
    G 0.3404255 0.1295238 0.3554604 0.4018059
    T 0.1829787 0.3142857 0.1927195 0.2460497
```

Bibliografía

Las referencias se listan por orden alfabético. Aquellas referencias con más de un autor están ordenadas de acuerdo con el primer autor.

[Aragón et al., 2016] Aragón, G. D., Arango, F. O., and Aranda, F. C. (2016). Cálculo del valor en riesgo operacional mediante redes bayesianas para una empresa financiera. *Contaduría y administración*, 61(1):176–201. [Citado en pág. xxi]

[Borovkov, 2013] Borovkov, A. A. (2013). Probability theory. [Citado en págs. 4 and 5]

[Chickering, 1995] Chickering, D. (1995). A new characterization of equivalent bayesian network structures. *Submitted for publication*. [Citado en pág. 40]

[Collins, 2021] Collins, F. S. (2021). Polimorfismos de nucleótido único (snps). https://tinyurl.com/8jch7k5c. [Citado en pág. 51]

[Dua et al., 2017] Dua, D., Graff, and Casey (2017). Uci machine learning repository. http://archive.ics.uci.edu/ml. [Citado en págs. xxi, 55, and 65]

[Friedman et al., 1999] Friedman, N., , Nachman, I., and Pe'er, D. (1999). Using bayesian networks to analyze whole-genome expression data. https://www.cs.huji.ac.il/labs/compbio/expression/html_ver/sld008.htm. [Citado en pág. 50]

[Friedman et al., 2000] Friedman, N., Linial, M., Nachman, I., and Pe'er, D. (2000). Using bayesian networks to analyze expression data. *Journal of computational biology*, 7(3-4). [Citado en págs. xxii and 49]

[G Towell and Shavlik, 1991] G Towell, M. N. and Shavlik, J. (1991). Primate splice junction gene sequences (dna) with associated imperfect domain theory. https://archive.ics.uci.edu/ml/datasets/Molecular+Biology+(Splice-junction+Gene+Sequences). [Citado en págs. xviii, xxi, xxii, 55, 61, 65, 83, and 85]

[Kjaerulff and Madsen, 2008] Kjaerulff, U. B. and Madsen, A. L. (2008). Bayesian networks and influence diagrams. *Springer Science+ Business Media*, 200:114. [Citado en pág. 31]

[Koller and Friedman, 2009] Koller, D. and Friedman, N. (2009). *Probabilistic graphical models: principles and techniques*. MIT press. [Citado en págs. xxi, xxi, xxi, xxi, 4, 13, 18, 19, 20, 21, 27, 31, 43, and 45]

[Lauritzen and Spiegelhalter, 1988] Lauritzen, S. L. and Spiegelhalter, D. J. (1988). Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society: Series B (Methodological)*, 50(2):157–194. [Citado en pág. 15]

[Margaritis, 2003] Margaritis, D. (2003). Learning bayesian network model structure from data. Technical report, Carnegie-Mellon Univ Pittsburgh Pa School of Computer Science. [Citado en pág. 38]

[Nagarajan et al., 2013] Nagarajan, R., Scutari, M., and Lèbre, S. (2013). Bayesian networks in r. *Springer*, 122:125–127. [Citado en págs. 9, 10, and 41]

[Puga et al., 2007] Puga, J. L., García, J. G., De la Fuente Sánchez, L., and De la Fuente Solana, E. I. (2007). Las redes bayesianas como herramientas de modelado en psicología. *Anales de Psicología/Annals of Psychology*, 23(2):307–316. [Citado en pág. xxi]

[Scutari, 2007] Scutari, M. (2007). Bnlearn r package. https://www.bnlearn.com/. [Citado en págs. 52 and 64]

[Scutari and Denis, 2014] Scutari, M. and Denis, J.-B. (2014). *Bayesian networks: with examples in R*. CRC press. [Citado en pág. 7]

[Spellman et al., 1998] Spellman, P. T., Sherlock, G., Zhang, M. Q., Iyer, V. R., Anders, K., Eisen, M. B., Brown, P. O., Botstein, D., and Futcher, B. (1998). Comprehensive identification of cell cycle–regulated genes of the yeast saccharomyces cerevisiae by microarray hybridization. *Molecular biology of the cell*, 9(12):3273–3297. [Citado en págs. 49 and 50]

[Su et al., 2013] Su, C., Andrew, A., Karagas, M. R., and Borsuk, M. E. (2013). Using bayesian networks to discover relations between genes, environment, and disease. *BioData mining*, 6(1). [Citado en págs. xxii, 37, 49, and 69]

[Tsamardinos et al., 2003] Tsamardinos, I., Aliferis, C. F., and Statnikov, A. (2003). Algorithms for Large Scale Markov Blanket Discovery. In *Proceedings of the Sixteenth International Florida Artificial Intelligence Research Society Conference*, pages 376–381. AAAI Press. [Citado en pág. 39]

[Wasserman, 2013] Wasserman, L. (2013). *All of statistics: a concise course in statistical inference*. Springer Science & Business Media. [Citado en págs. 4 and 6]