Un algoritmo para el descubrimiento automático de prerrequisitos curriculares.

An algorithm for automatic discovery of curricular prerequisites.

Tema: Inteligencia Artificial. Modelado y Análisis de Datos.

Trabajo de Fin de Grado en Matemáticas

Autor: **Sara Ruiz Ruiz**

Departamento: **Lenguajes y Ciencias de la Computación.**

Número de páginas (sin incluir introducción, bibliografía, ni anexos): **34**

Junio 2020

DECLARACIÓN DE ORIGINALIDAD DEL TFG

Dña. *Sara Ruiz Ruiz*, con DNI *76877939W*, estudiante del Grado en *Matemáticas* de la Facultad de Ciencias de la Universidad de Málaga,

DECLARO:

Que he realizado el Trabajo Fin de Grado titulado "*Un algoritmo para el descubrimiento automático de prerrequisitos curriculares.*" y que lo presento para su evaluación. Dicho trabajo es original y todas las fuentes bibliográficas utilizadas para su realización han sido debidamente citadas en el mismo.

Para que así conste, firmo la presente en Málaga, el *10 de julio de 2020.*

Dedicado a mi familia,
por apoyarme y creer en mí
en todas aquellas veces que yo desistía.

A Alicia,
por ser pura definición de amistad
y compañerismo durante cuatro años.

A Rocío, al resto de compañeros matemáticos
y a todos mis amigos del Erasmus,
por hacer del camino algo mucho más disfrutable.

Agradecimientos

Con este trabajo de fin de grado acabo mi larga etapa como estudiante de matemáticas. No de la mejor forma ni de la esperada, pero gracias a los profesores, a la comunidad universitaria y a la era tecnológica en la que vivimos, ha sido posible trabajar desde casa para llevar este proyecto adelante. Agradezco haber sido capaz de acabar mis estudios aún encontrándonos en una situación tan difícil para el mundo como ha sido la pandemia por el COVID-19.

Gracias a la Universidad de Málaga, por permitirme que este TFG sea parte de un trabajo de investigación. Éste está ligado a un proyecto del plan nacional (Plan Propio de Investigación y Transferencia) y al proyecto SIETTE (www.siette.org).

Por último, gracias a Beatriz Barros, tutora de este trabajo. Por su afán trabajador y su paciencia al enseñar desde cero.

Índice general

Un algoritmo para el descubrimiento automático de prerrequisitos curriculares.

Resumen

En educación, las unidades de conocimiento que se presentan a un alumno, también llamadas temas, están secuenciadas según su dificultad. Además, un diseño razonable del proceso de enseñanza debe tener en cuenta una adecuada organización de los contenidos, considerando la necesidad del conocimiento previo de algunos de ellos para comprender los siguientes; esto es, se deben tener en cuenta las dependencias de prerrequisitos existentes entre los diferentes temas.

En este trabajo de fin de grado, se estudian las posibilidades de utilizar métodos computacionales para generar automáticamente la estructura de dependencias de prerrequisitos entre temas y subtemas de una materia; con la ayuda de grandes cantidades de datos sobre evaluaciones parciales de estudiantes. Para ello, se ha diseñado e implementado en Python un algoritmo que se ha probado con datos de alumnos reales. En el trabajo se usan además, métodos probabilísticos y estructuras propias de la Inteligencia Artificial.

Como resultado, se genera como salida un modelo representado como una red bayesiana que (i) puede ayudar al profesor a una secuencia óptima de presentación de los conceptos; (ii) o ser utilizada como referencia en un un sistema de enseñanza y aprendizaje para adaptar la secuenciación de conocimientos a las necesidades de cada alumno.

Palabras clave:

Inteligencia Artificial, Aprendizaje automático, Descubrimiento de Prerrequisitos, Red Bayesiana, Minería de Datos

An algorithm for automatic discovery of curricular prerequisites.

Abstract

In education, knowledge units presented to students, are sequenced according to their difficulty. Also, a reasonable teaching process design must take into account a proper organization of the contents, considering the need of prior comprehension of some of them to comprehend the following ones; that is, they must take into account the existing prerequisites dependencies between the units themselves.

In this final degree project, the possibilities of using computational methods to generate automatically such a prerequisites structure between units and subunits, are studied; with the help of large amounts of data on partial student evaluations. To do so, an algorithm in programming language *python* has been designed, implemented and tested with real student data. In the project are used as well, Probabilistic methods and classic structures of Artificial Intelligence.

As a result, a model represented as a bayesian network is generated as output. This way, (i) it can help the professor generate an optimal sequence of the concepts for their adquirement or (ii), it can be used as reference in a teaching and learning system for an adaptation of the knowledge sequence varying to the needs of each student.

Key words:

Artificial Intelligence, Machine learning, Prerequisite Discovery, Bayesian Network, Data Mining

Capítulo I

Introducción

En este Trabajo de Fin de Grado se implementa un algoritmo que permite analizar resultados de alumnos a pruebas de evaluación de contenidos variados. El objetivo final del trabajo es estudiar la posibilidad de extraer conclusiones sobre las relaciones existentes entre tales contenidos a partir de la estadística de los resultados de un número -en el caso ideal- suficientemente alto de alumnos.

Empezaremos por exponer brevemente en qué consisten estas relaciones y profundizaremos un poco más en el planteamiento de nuestro problema.

I.1. Estructuras de Aprendizaje: Los Prerrequisitos

En todos los entornos de aprendizaje, la información que se facilita a los estudiantes está ordenada de alguna manera; normalmente tratando de moverse gradualmente de lecciones más simples a otras más complejas. En este proceso, habrá conceptos necesarios de aprender y asimilar antes de poder comprender uno posterior [7].

Denominamos **estructura de prerrequisitos** al conjunto de relaciones entre conocimientos que implican orden óptimo a la hora de ser adquiridos. Asimismo, un concepto A se denomina **prerrequisito** de otro concepto B si es altamente recomendable (y muchas veces, incluso necesario) aprender A antes que B para una buena y completa comprensión del segundo.

Un claro ejemplo se puede ver en el aprendizaje temprano de las matemáticas [7]: Para que un niño aprenda a multiplicar, es necesario que conozca y domine antes la operación suma. Así pues, en este caso sencillo y, sin lugar a debate, se dice que la operación suma es prerrequisito de la multiplicación, y por ello debe enseñarse antes en las aulas.

I.2. Planteamiento del problema

Nos planteamos el origen de tal orden de adquisición de conocimientos impuesto en cada uno de los entornos de aprendizaje que conocemos, esto es, nos planteamos quién y

en base a qué criterio ha definido implícitamente cada una de las estructuras de prerrequisitos. La mayoría han sido el resultado de un gran esfuerzo de expertos en la materia que han determinado las relaciones entre los conceptos según experiencia propia. Pero este proceso puede ser muy largo y además no siempre tiene buenos resultados. En los últimos años, se plantea la posibilidad de la creación de estas estructuras de prerrequisitos automáticamente a través de grandes cantidades de datos que contienen información sobre resultados de evaluación de alumnos a cuestiones, conceptos y problemas.

I.3. Objetivos

En este contexto, este trabajo pretende desarrollar un algoritmo de forma que, a partir de gran cantidad de datos de una materia (organizada en temas y subtemas), se puedan modelar las estructuras de aprendizaje de forma automática. Además, de esta manera se permite modelar tareas que se presentan a alumnos teniendo en cuenta su nivel cognitivo, la evolución de su aprendizaje y la información de contexto, que se infiere a partir de datos de estudio de la materia de alumnos anteriores. Esta investigación se enmarca en el área de la Inteligencia Artificial y Educación (AIED), el Análisis de Datos y el Desarrollo de Sistemas.

Una vez descrito el algoritmo, se plantea comprobar la eficacia y el correcto funcionamiento de éste. Para ello nos ayudamos de un conjunto de datos obtenido de usuarios reales, de la plataforma SIETTE, de la cual hablaremos en el epígrafe 1.5. El conjunto proporciona gran cantidad de datos de alumnos de la Universidad de Málaga contestando a pruebas que abarcan cinco diferentes temas de contenido. Se sacarán conclusiones sobre los prerrequisitos detectados en este caso; aunque el propio algoritmo se diseña para funcionar con un conjunto de datos cualquiera.

Resumiendo, este TFG tiene como objetivo explorar la posibilidad de usar el algoritmo para que de manera automática y eficaz, obtengamos una óptima estructura de prerrequisitos entre un conjunto de temas que conforman un dominio, a partir de evaluaciones parciales de cada tema y subtema de un conjunto de alumnos que cursaron esa asignatura.

I.4. Fases del Trabajo

Supongamos tener ante nosotros un conjunto de datos que cumpla y posea la siguiente información:

- Se recogen datos que de alguna manera evalúen los conocimientos de alumnos en dos o más temas de conocimiento.
- Se conoce qué resultado corresponde a qué tema (idealmente una cuestión correspondería unívocamente a un tema concreto, pero esto no siempre es así).

Para llevar a cabo el objetivo principal sobre tal conjunto de datos, el trabajo se divide en las siguientes fases:

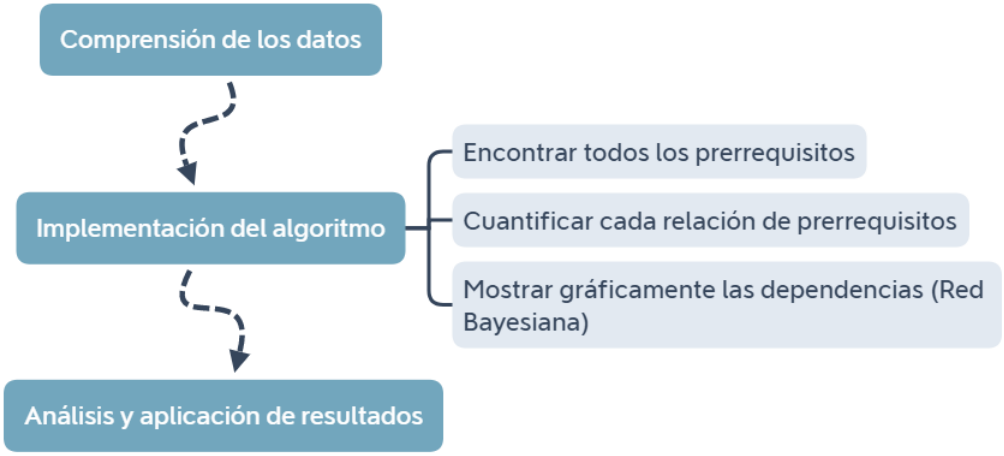


Figura I.1: Fases del Trabajo

1. *Comprensión y Preparación de los datos.* A partir del conjunto de datos, es necesaria la comprensión de la información que presenta. En ese caso, se han de discretizar los datos a un nuevo fichero que nuestro algoritmo acepte como fichero de entrada y sea capaz de trabajar con él.

2. *Definición e Implementación de un algoritmo que genere los prerrequisitos con los datos disponibles.* El algoritmo, a partir del fichero de entrada, obtiene la estructura de prerrequisitos óptima entre los temas modelando la salida en forma de red bayesiana. Esta fase incluye a su vez todos los sub-objetivos que esperamos que cumpla el propio algoritmo desarrollado:

   a) Búsqueda de todos los prerrequisitos entre temas.

   b) Cuantificación de la importancia de cada uno de los prerrequisitos encontrados.

   c) Representación gráfica de todas las relaciones encontradas (Esto se va a hacer gracias a la exposición de los resultados en una red bayesiana)

3. *Análisis y aplicación los resultados para generalizarlos en el dominio de los datos.* Por último, se espera un análisis y conclusión de los resultados obtenidos. A partir de la red bayesiana obtenida, se puede establecer un orden óptimo en la estructura de aprendizaje de la materia en cuestión; o incluso, modelar vías de aprendizaje adaptadas a cada alumno según sus propias evaluaciones.

I.5. Organización de la Memoria.

La presente Memoria se ha estructurado en cuatro capítulos:

- El primer capítulo (**Contexto del trabajo**) comienza exponiendo otros métodos que abarcan el problema. A continuación, se define y se justifica nuestra elección.

Además, se introduce información sobre el software y programas sobre los que el trabajo se apoya.

- En el segundo capítulo (**Algoritmo para la detección de prerrequisitos**) se expone desde un enfoque teórico el funcionamiento de un primer algoritmo; ello permite la búsqueda de posibles errores, que se resuelven adecuadamente. Finalmente, se muestra su funcionamiento real con un ejemplo.
- El tercer capítulo (**Aplicación del algoritmo a datos reales**) está dedicado a la aplicación del algoritmo definido anteriormente a datos reales de pruebas de evaluación de nivel universitario. Los datos usados forman parte de la asignatura *Estructuras Discretas* del grado de Informática impartido en la UNED. Se exponen los resultados y se comentan brevemente.
- Por último, en el cuarto y último capítulo (**Conclusiones y líneas de investigación futuras**), se analizan todos los procesos estudiados y todo el trabajo realizado. Se muestran las ventajas que nuestro algoritmo proporciona frente a otros métodos y se proponen vías de futuro avance.

Además, el trabajo cuenta con un anexo (**Programación en Python**) en el que se recoge todo el código programado para llevar a cabo el TFG.

Capítulo 1

Contexto del Trabajo

1.1. Métodos existentes para la resolución del problema

Este tema de investigación es clásico en el campo de la generación y estructuración de conocimiento, y ya varios investigadores han desarrollado algunos métodos automáticos para extraer estructuras de prerrequisitos a partir de datos en las últimas décadas, tanto para enseñanza como para clasificación de conceptos. Se pueden establecer dos grades categorías según el tipo de datos que consideran [5]: aquellos que tienen en cuenta el resultado final del proceso de enseñanza y aprendizaje (respuestas de los alumnos generalmente) y aquellos que tienen en cuenta el contenido (figura 1.1). A su vez, éstos se subdividen según la manera de enfocar el problema y el método usado para resolverlo. En esta sección, se mencionará brevemente de qué trata cada uno de los métodos ya existentes.

En primer lugar, como las relaciones de prerrequisitos son un tipo de relación causal y además los conocimientos pueden ser modelados como variables, existe un método que generaliza los algoritmos de descubrimiento de estructuras causales. Ello supone conocer la matriz-Q que especifica qué ítems miden qué conocimientos, y a partir de dicho modelo busca los prerrequisitos con los algoritmos de descubrimiento causales [10].

Otro método es el usado en [7] para obtener los prerrequisitos; en primer lugar usa el algoritmo de Maximización de la Esperanza Estructural para seleccionar una clase de redes bayesianas equivalentes y posteriormente usa información adicional para seleccionar una única red. De forma similar, podemos usar diferentes estimaciones bayesianas que nos ayuden a determinar la estructura de prerrequisitos. En [11], se usan métodos de Monte Carlo basados en cadenas de Markov para estimar y más tarde se comparan modelos anidados.

En [2], estudian el problema con una Evaluación de las hipótesis. La idea es estimar probabilidades, dadas distintas posibles estructuras de prerrequisitos. Además, el mismo proyecto estudia la posibilidad de obtener la estructura a partir de observaciones con ruido experimental, que es el caso de muchos escenarios reales.

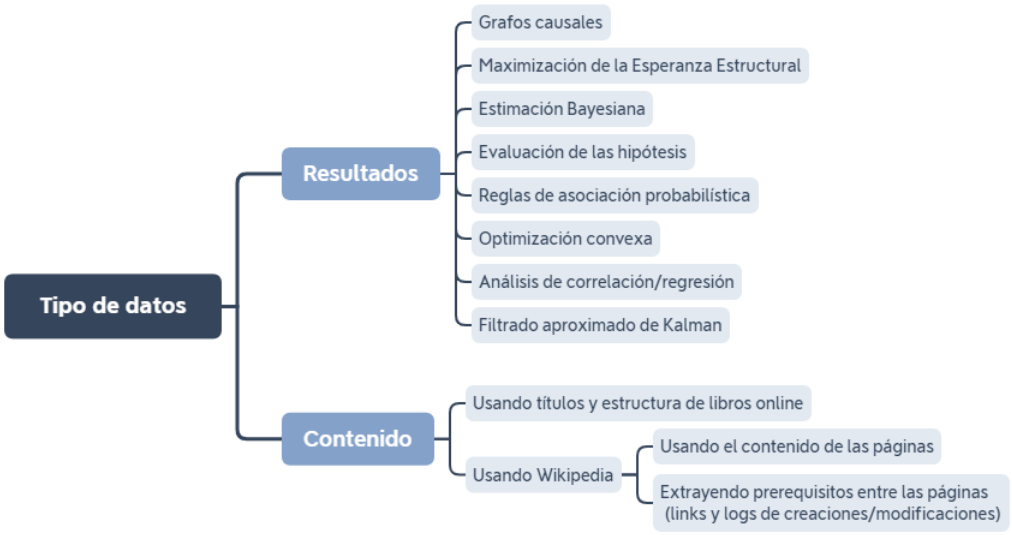


Figura 1.1: Tipos de datos asociados al problema

Un método diferente ya usado es preprocesar los datos de resultados de alumnos, gracias a un modelo de evidencia, y usarlo para descubrir la estructura de prerrequisitos de conocimientos. Este método se adapta a los datos de prueba y de registro con diferentes modelos de evidencia [6]. Más métodos incluyen optimización convexa [15], análisis de correlación/regresión [4] y filtrado aproximado de Kalman [13].

Por otro lado, tenemos los métodos que usan datos del contenido de la materia en vez de resultados a ítems resueltos por los alumnos. Por ejemplo, [12], usa los títulos y el orden de libros online para modelar la estructura de prerrequisitos. Muchos otros trabajos usan datos de Wikipedia: tanto el contenido de sus páginas como las relaciones entre ellas. Este último es el caso de [14], que usa los links entre páginas como relaciones entre conceptos y así presenta su estructura.

En nuestro trabajo, utilizaremos como principal recurso para obtener los resultados acerca de la estructura de requisitos de una materia determinada, un estudio estadístico. Además se hace uso de las Redes Bayesianas para la representación del resultado. Se propone este mecanismo de modelado de los prerrequisitos porque (1) esta estructura, que se describe en el apartado siguiente, se ajusta al concepto de vinculación de dependencias, como ocurre en los prerrequisitos; y (2) porque se quiere estudiar si es posible generar de forma automática una red de este tipo con los datos de evaluación procedentes del sistema *siette* (que se describe en la sección 1.5)

1.2. Una introducción a las Redes Bayesianas

Para la construcción formal de una red bayesiana, partimos inicialmente de un número finito de nodos e introducimos unas definiciones previas necesarias. Con ello podremos posteriormente definir formalmente el concepto de redes bayesianas, siguiendo el mismo esquema seguido por Francisco Díez en [9]:

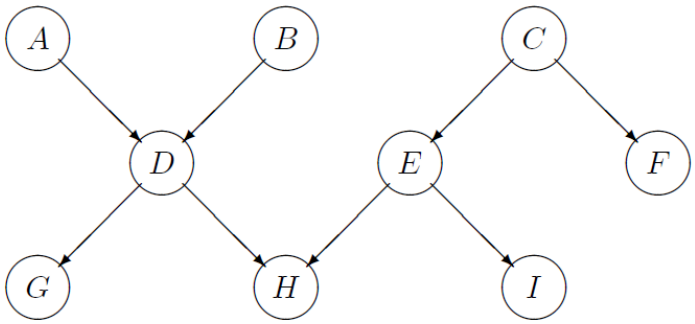


Figura 1.2: Representación de un grafo dirigido. Ejemplos: D es hijo de A y padre de G. H es descendiente de C [9]

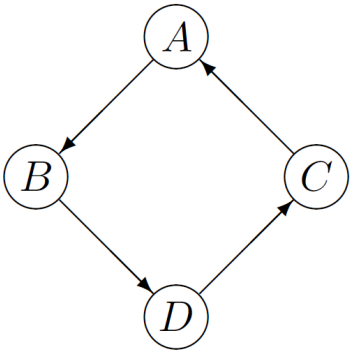


Figura 1.3: Ejemplo de ciclo [9]

- **Arista direccional**. Es un par ordenado de nodos (X, Y). En la representación gráfica viene dado por una flecha desde X hasta Y.
- **Grafo dirigido**. Es el par $(\mathcal{N}, \mathcal{A})$ donde $\mathcal{N}$ es un conjunto de nodos y $\mathcal{A}$ un conjunto de aristas direccionales sobre los nodos.
- **Nodo padre/hijo**. El nodo X es padre de Y si y sólo si existe una arista direccional (X,Y). En este caso se dice que Y es nodo hijo de X.
- **Descendiente**. Sea $(\mathcal{N}, \mathcal{A})$ un grafo dirigido y sean $X, Y \in \mathcal{N}$, se dice que X es descendiente de Y si existe un conjunto de nodos $X_1, ..., X_n \in \mathcal{N}$ tales que las aristas direccionales $(Y, X_1), (X_i, X_{i+1}), (X_n, X) \in \mathcal{A} \quad \forall i \in \{1, ..., n-1\}$.
- **Ciclo**. Es una sucesión de nodos $\{X_1, ..., X_N\}$ pertenecientes a un grafo, distintos entre sí y tales que para todo $i \in \{1, ..., N-1\}$ existe la arista direccional (X_i, X_{i+1}), además de existir la arista (X_N, X_1).

Una **red bayesiana** es una representación en forma de grafo acíclico dirigido de un modelo probabilístico. Se compone de nodos, aristas direccionales y de una distribución de probabilidad sobre sus variables. El conjunto finito de **nodos** representa variables aleatorias, como por ejemplo el resultado de la tirada de un dado, una condición médica, etc. y pueden tomar valores discretos o continuos. Esta identificación entre nodos y variables es biunívoca, luego las llamaremos indistintamente y serán representadas de ahora en adelante por letras mayúsculas. Además, vamos a seguir la siguiente notación: Dado un nodo X, que puede tomar diferentes valores según su definición, vamos a representar por x minúscula a un valor cualquiera de dicho nodo.

Por otro lado, las **aristas** direccionales representan dependencia directa entre las variables aleatorias que unen. Si no aparece una arista entre dos determinados nodos, significará por tanto, que hay independencia entre ellos. Además, a cada variable aleatoria se le asocia una **distribución de probabilidad**, con la única condición de que cumpla la propiedad de *separación direccional*.

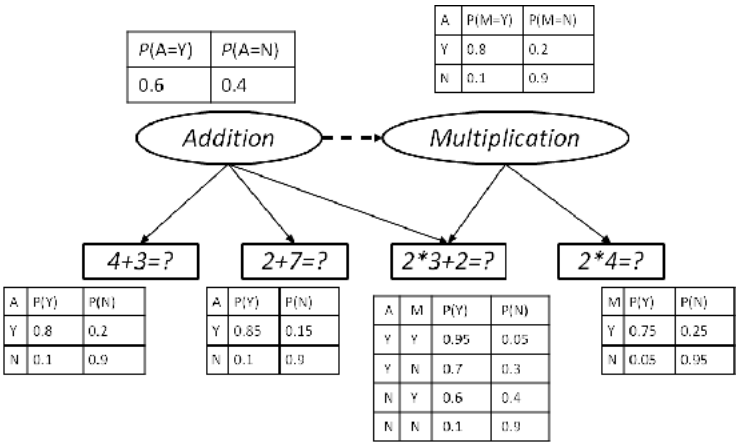


Figura 1.4: Hipotética Red Bayesiana [7]

- **Separación direccional**. Dado un grafo acíclico dirigido con una distribución de probabilidad sobre sus variables, se dice que cumple la propiedad de separación direccional si:

  Para todo nodo X, el conjunto de sus padres $pa(X)$, y otro subconjunto de nodos $\tilde{Y}$ que no contenga descendientes de X se tiene que $P(x \mid pa(x), \tilde{y}) = P(x \mid pa(x))$.

Observación: En la definición anterior, por semejanza a la notación usada para los nodos y sus valores, $pa(x)$ representa un vector formado por valores cualesquiera de los padres de X e $\tilde{y}$ representa un vector formado por valores cualesquiera del conjunto de nodos $\tilde{Y}$.

Ya estamos en condiciones de definir formalmente una red bayesiana:

- **Red bayesiana**. Es un grafo acíclico dirigido con una distribución de probabilidad asociada a sus variables que cumple la propiedad de la separación direccional.

Esto es, para cada nodo, necesitamos saber primeramente cuáles son sus nodos padres y todas sus posibles combinaciones de valores. Dada cada una de las combinaciones, tenemos las probabilidades del nodo en cuestión. De esta forma, sabemos la probabilidad de que dicha variable aleatoria tome un valor determinado, dado el valor de sus nodos padres. En la práctica, entonces, la información cuantitativa de la red bayesiana viene dada por la probabilidad a priori de los nodos que no tienen nodos padres y por la probabilidad condicionada de los nodos que sí tienen padres (probabilidades que encontramos en las tablas) y la información cualitativa viene dada por la estructura del grafo en sí.

Para una mejor comprensión, en la figura 1.4 se muestra cómo sería una red bayesiana con el ejemplo mencionado anteriormente de las operaciones suma y multiplicación [7]: Tenemos dos nodos; las aristas sólidas relacionan elementos (cuestiones de tests) a la habilidad que se necesita conocer para superarla con éxito y las aristas discontinuas son las definidas anteriormente y que deben ser descubiertas gracias a los datos. El nodo suma se observa que es nodo padre de multiplicación. Encima de cada nodo vemos las tablas de probabilidades condicionadas.

Una de las ventajas de las redes bayesianas es la eficiencia al calcular probabilidades a posteriori, que es la llamada inferencia probabilística. Estas probabilidades a posteriori

son las definidas por el teorema de Bayes:

$$P(A \mid B) = \frac{P(B \mid A)\, P(A)}{P(B)}$$

donde $P(A)$ es la probabilidad a priori (y sería el nodo padre del que sabemos previamente las probabilidades), $P(B \mid A)$ lo obtendríamos de la tabla de probabilidades condicionadas; y por último $P(B)$ es la evidencia, es decir, la probabilidad obtenida al contar los casos en los que ocurre B de entre todos los casos en general. De esta forma, nos es fácil obtener la probabilidad de las causas dados los efectos, cuando normalmente los datos que tenemos son los contrarios.

A menudo, la propia red bayesiana también se usa para ajustar o aprender parámetros desconocidos. Otro ejemplo sería el aprendizaje automático de la estructura de la red cuando esta tarea es demasiado compleja para el ser humano. Para que todo esto se haga con éxito es necesaria una red de grandes dimensiones

1.3. Redes Bayesianas y Prerrequisitos

En trabajos de los últimos años, se ha observado el interés y el éxito del uso de redes bayesianas en diferentes áreas. En referencia al aprendizaje y a las estructuras de prerrequisitos, el trabajo realizado en [3], expone como principal resultado la importancia de dicha estructura para un buen modelado estudiantil.

A partir de una red bayesiana que tenía en cuenta la actitud y los conocimientos del alumnado, se creó un curso online de Logic. Surgió de inmediato el problema de no haber tenido en cuenta la inclusión de las relaciones de prerrequisitos pues éstas no sólo sirven como guía al tomar acciones educativas (secuencia de temas, generar ejercicios adecuados, etc.) sino también son una muy útil herramienta para conocer el estado de conocimiento de cada estudiante. En este mismo trabajo, se analiza la validez y la mejora de los resultados al añadir una nueva capa al modelo con la inclusión de una red bayesiana con la estructura de prerrequisitos. Hay dos grandes observaciones: la primera, que el incluir la estructura de prerrequisitos usando redes bayesianas es un acertado enfoque pues simula el razonamiento humano en este contexto adecuadamente. Y además, en segundo lugar, se observa con éxito cómo este paso mejora la eficiencia del proceso de diagnóstico, permitiendo así una mayor exactitud en los resultados.

1.4. El sistema SIETTE

En el proceso de enseñanza y aprendizaje (en cualquier ámbito), siempre ha tenido hueco la evaluación del alumnado, que ayuda a los mentores a mejorar y conocer el nivel de la clase y a los alumnos a conocer de forma objetiva si están siguiendo el ritmo.

Una de las formas más comunes de realizar esta evaluación es el uso de pruebas automatizadas [8] modeladas como un conjunto de preguntas con formatos diferentes, que van más allá de la elección de una lista de opciones. Esto permite realizar exámenes de forma automática. La realización de pruebas automatizadas es cómoda para el profesor,

Figura 1.5: Ventana principal del sistema siette (www.siette.org)

pues tiene fácil y rápida corrección. Además es una forma de corrección sistematizada y totalmente objetiva dónde ningún alumno sale beneficiado ni perjudicado. Pero este método (en su forma clásica) también tiene inconvenientes: Uno de ellos es que no se tiene en cuenta la diferencia de dificultad entre las preguntas del mismo, o la probabilidad de acertar algunas preguntas por azar y no por conocimiento de la materia. Este problema se ha intentado paliar, surgiendo así la teoría de la respuesta al ítem (1960-1970). Otro inconveniente es la no adaptación del test al propio alumno, ya que en general son test idénticos para alumnos diferentes, de solucionar esto se ha encargado la teoría de tests adaptativos informatizados (principios de los 1980), que al aplicar tests a través de ordenadores, facilita el modelado de las preguntas según quién esté al otro lado de la pantalla y su proceso de aprendizaje.

SIETTE, Sistema Inteligente de Evaluación mediante Test para TeleEducación [8], es una plataforma que recoge los resultados de ambas teorías para la realización de pruebas de aprendizaje online (1.5).

La forma de funcionar de SIETTE consiste en un módulo de edición que permite a los profesores crear una estructura del temario, almacenar preguntas y respuestas y especificar qué temas quiere evaluar. Todo esto se almacena en una base de conocimientos, donde se encuentra la colección de posibles preguntas, todas ellas calibradas según dificultad, temario al que evalúan y múltiples parámetros. Lo siguiente, es el módulo generador de pruebas que selecciona preguntas basándose en ambas teorías planteadas anteriormente para modelar un test a cada alumno en cada momento que se adapte a todas sus necesidades y tenga en cuenta la dificultad de las preguntas. Por otro lado, si un profesor crea personalmente un test, SIETTE también se encarga de validar sus datos y asegurarse de su consistencia antes de ponerlo a disposición del alumnado. Por último, entra en juego el módulo de aprendizaje, que se encarga continuamente de recalibrar los parámetros de las preguntas según los datos obtenidos por las respuestas de los alumnos.

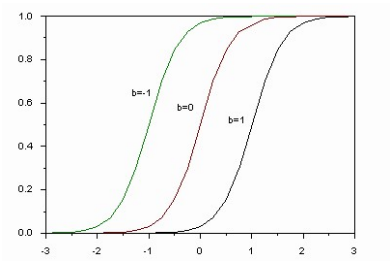


Figura 1.6: diferentes dificultades del ítem

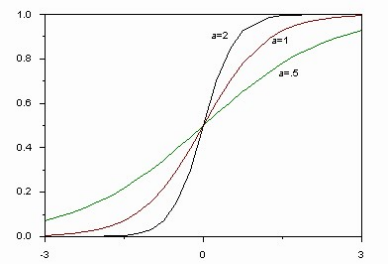


Figura 1.7: diferentes factores de discriminación

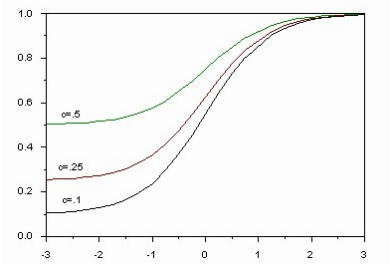


Figura 1.8: diferentes factores de adivinanza

Ejemplos de CCI [1]

Debido a que en SIETTE el modelo de usuario está basado en la Teoría de la Respuesta al Ítem mayoritariamente, vamos a explicar brevemente en qué consiste ésta. La teoría se basa en que la respuesta de un alumno a un ítem depende probabilísticamente de un cierto valor numérico llamado rasgo, en nuestro caso, este rasgo sería el nivel de conocimiento del alumno en la materia que se está examinando. Este rasgo es en un principio una variable desconocida y vamos a suponer que puede medirse con un número real dado entre $(-\infty, +\infty)$. Suponemos conocida, para cada ítem i-ésimo, una función θ_i definida en $(-\infty, +\infty)$ y que toma valores en $[0, 1]$, que nos indica que la probabilidad de un alumno con rasgo (nivel de conocimiento) x de acertar el ítem i-ésimo es $\theta_i(x)$. Como es de esperar, esta función llamada Curva Característica del Ítem (CCI), dependerá de la dificultad de la cuestión (fig 1.6); pero también tiene en cuenta el factor de adivinanza (probabilidad de que un alumno responda correctamente al azar, fig 1.8) y el factor de discriminación (proporcional a la pendiente de la curva, cuanto mayor es, mayor es la probabilidad de un alumno con un nivel de conocimiento mayor al nivel de dificultad de la pregunta de acertar, fig 1.7). Una observación sencilla es que la CCI debe ser monótona creciente pues cuanto mayor es el nivel de conocimiento de un alumno, mayor debe ser su probabilidad de acertar el ítem. Algo que nos debe sorprender entonces es: hemos definido una función θ_i que depende del valor de la variable rasgo de un determinado alumno, pero este rasgo hemos dicho que es desconocido en un principio. Pues bien, éste se estima según el alumno responde a cuestiones, y hay diferentes métodos para hacerlo, como el método de la máxima verosimilitud, método bayesiano, método basado en redes neuronales, etc. Un breve comentario sobre estos métodos lo encontramos en [8].

Por otro lado, hemos mencionado que SIETTE también se basa en la Teoría de Test Adaptativos con el fin de plantear pruebas de evaluación adaptados al nivel de conocimiento del alumno y así que éste ni se aburra ni se frustre en el camino. El procedimiento que sigue es estimar primeramente el rasgo del alumno y presentarle un ítem acorde con dicho nivel de conocimiento. A partir de esta primera respuesta se produce una nueva estimación del nivel de conocimiento y se vuelve a plantear una cuestión adaptada a dicho nivel. Se trabaja así sucesivamente hasta que se cumpla un cierto criterio de finalización.

SIETTE es entonces una buena plataforma para la evaluación del nivel de conocimiento del alumnado y es por ello que la usaremos para obtener los datos masivos que necesitamos para llevar a cabo nuestro trabajo de investigación.

1.5. GeNIe Modeler: Herramienta para representar Redes Bayesianas

Como ya hemos mencionado anteriormente, para este trabajo hemos decidido usar redes bayesianas como la herramienta principal para representar nuestros modelos probabilísticos obtenidos una vez aplicado el algoritmo. Debido a su gran uso, encontramos numerosas opciones disponibles de software para dicha representación, dependiendo del tipo de usuario y sus necesidades. En nuestro caso, para la representación y una fácil implementación de cualquier tipo de inferencia estadística, vamos a hacer uso de *GeNIe Modeler*, una herramienta diseñada para modelados de inteligencia artificial y aprendizaje automático con redes bayesianas y otros tipos de modelos probabilísticos gráficos. Es un producto comercial que lleva en continua actualización desde 1999 y ha sido diseñada para Windows por BAYESFUSION, LCC. La interfaz gráfica que posee es versátil y de fácil manejo [16].

El programa GeNIe Modeler es una interfaz gráfica de usuario (GUI) que facilita la visualización de los resultados de las redes bayesianas y permite aprendizaje y modelado interactivo del propio modelo. Los nodos de la red se representan con círculos que contienen una etiqueta de reconocimiento (un nombre) y las aristas direccionales con flechas entre los nodos. Al pulsar sobre un nodo cualquiera de la red; aparece una ventana de información sobre él. Si estamos construyendo la red bayesiana al momento, nos permite definir la tabla de probabilidades condicionadas correspondiente al nodo según los padres. Si la red está ya construida, nos permite ver la tabla y modificarla. El propio software pone límites para que la red esté bien definida; esto es, no permite cualquier disposición de aristas pues esto podría originar bucles. Además, a la hora de definir las tablas de probabilidades condicionadas, sólo permite hacerlo de tal manera que las leyes de probabilidad se cumplan (probabilidades sólo toman valores entre 0 y 1, probabilidades complementarias suman el valor completo de 1, etc.)

Ahora, para nuestra aplicación, nos planteamos la posibilidad de que a través del lenguaje Python y de los datos obtenidos se pueda generar la red en GeNIe; de esta manera, el código permitiría la aparición automática de la red. Esto es posible gracias a la librería *SMILE Engine*, concretamente a *PySmile*, la adaptación a Python. Ésta logra la creación de un fichero tipo GeNIe Network (.xdsl) con los nodos, aristas y modelo probabilístico definido en un fichero python ejecutable.

Siguiendo el ejemplo sencillo que nos ayudará a lo largo de todo el documento de las operaciones suma y multiplicación; el siguiente código escrito en Python nos muestra cómo se puede crear un fichero GeNIe de una red bayesiana. En la figura 1.9 se observa el resultado obtenido al ejecutar el código.

```
1 import pysmile
2 import pysmile_license
3
4 class RedBayesianaGenie:
5
6     def __init__(self):
7         net = pysmile.Network()
8
```

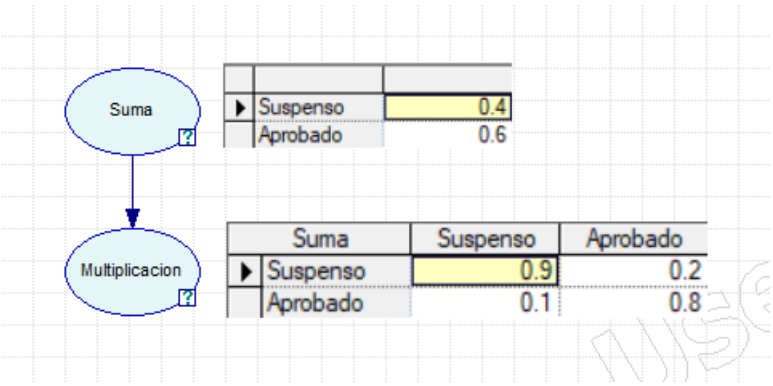


Figura 1.9: Red bayesiana obtenida en el software GeNIe Modeler al ejecutar el código Python descrito

```
9           self.create_cpt_node(net,"Suma","Suma",["Suspenso","Aprobado"
                ],60,40)
10          self.create_cpt_node(net,"Multiplicacion","Multiplicacion",["
                Suspenso","Aprobado"],60,140)
11
12
13          net.add_arc("Suma","Multiplicacion");
14
15          tabla1 = [
16          0.4, # P(Suma=Suspenso)
17          0.6, # P(Suma=Aprobado)
18          ]
19          net.set_node_definition("Suma",tabla1)
20
21          tabla2 = [
22          0.9, # P(Multiplicación=Suspenso | Suma=Suspenso)
23          0.1, # P(Multiplicación=Aprobado | Suma=Suspenso)
24          0.2, # P(Multiplicación=Suspenso | Suma=Aprobado)
25          0.8, # P(Multiplicación=Aprobado | Suma=Aprobado)
26          ]
27          net.set_node_definition("Multiplicacion",tabla2)
28
29          net.write_file("suma-mult.xdsl")
30
31
32  ejecuto = RedBayesianaGenie()
```

1.6. Plan de Trabajo

Se plantea combinar el estudio experimental realizado a través de programación de algoritmos en el lenguaje Python con los desarrollos teóricos que fundamentan éstos. La investigación se basa principalmente en el área de la Informática, más concretamente, en la Inteligencia Artificial. Aún así, se intenta enfocar siempre también desde un punto de vista matemático; esto es, intentando dar definiciones y demostraciones rigurosas a la hora de trabajar el marco teórico.

La metodología usada durante el trabajo para la implementación del código sigue el modelo de un proceso iterativo. En este contexto; el trabajo comienza por resolver casos simples y sencillos del mismo problema y mediante un proceso de depuración por prueba y error, se van resolviendo los problemas que surgen en el camino. Según se avanza con planteamientos de problemas más complejos, irán apareciendo problemas nuevos, cuya solución irá fortaleciendo las capacidades y robustez del algoritmo que se desarrolla. Este proceso se podría repetir tantas veces como fuera necesario hasta encontrar un algoritmo que funcione y no presente errores. Por ello; en los próximos capítulos se comenzará exponiendo un primer algoritmo para el caso sencillo de datos generados aleatoriamente y sus problemas. Se buscarán soluciones, se implementarán y posteriormente se pasará a probar el algoritmo con los datos reales, que es el caso más complejo.

Capítulo 2

Algoritmo para la detección de prerrequisitos

Como ya se ha mencionado anteriormente, nuestro trabajo de investigación pretende la búsqueda de prerrequisitos entre temas de una cierta materia a partir de un fichero con gran cantidad de datos (obtenido de la plataforma SIETTE). Los datos contienen información sobre los resultados de diferentes alumnos respondiendo diferentes tests sobre esos temas.

Para llevar a cabo este trabajo vamos a desglosar las tareas e ir avanzando una a una. En una primera toma de contacto crearemos un algoritmo que analice los prerrequisitos existentes en un fichero de datos generados aleatoriamente y que a partir de los resultados produzca automáticamente una red bayesiana que nos muestre visualmente tales dependencias entre temas y las probabilidades asociadas. Todo el proceso se hará de manera genérica para que pueda usarse posteriormente con los datos reales, nuestro objetivo principal.

2.1. Primer algoritmo para detectar Prerrequisitos dos a dos

El primer intento de algoritmo es sencillamente una aplicación paso a paso de la teoría estudiada. Este algoritmo encuentra en su proceso errores, al no haber tenido en cuenta posibles casos que se pueden dar en los datos y dificultan la situación. La corrección del algoritmo para que tenga en cuenta estas posibles situaciones de error hará que las sucesivas versiones del algoritmo consigan de forma robusta detectar los prerrequisitos existentes en cualquier posible escenario de datos.

2.1.1. Etapas del algoritmo

El algoritmo consta de las siguientes etapas (ver figura 2.1): Se generan los datos aleatorios con los que trabajar, se discretizan los datos para su fácil manejo y, posteriormente, se crea un programa que halle los prerrequisitos entre temas atendiendo a un método es-

Figura 2.1: Etapas del algoritmo

cogido en particular, así como las tablas de probabilidades condicionadas que se necesitan para definir una red bayesiana correctamente. A partir de estos datos, simplemente resta escribirlos en un fichero que el software de GeNie sea capaz de leer para que éste nos cree la red correspondiente.

- Generador de datos aleatorios

  Primeramente, se crea un fichero con números aleatorios de la siguiente manera: Elegimos cuántos alumnos (n) y cuántos temas (m) queremos simular. Ante esta información, se generan un total de nm respuestas. Dichas respuestas son un número real comprendido entre -1 y 1 que quieren representar el conocimiento del alumno sobre el tema en cuestión, teniendo en cuenta que el -1 corresponde a un conocimiento nulo y el 1 a un conocimiento total. Así pues, los datos son generados con la librería *random* de Python y son inmediatamente escritos en un fichero excel. El comando *uniform(a,b)* genera un número float aleatorio entre a y b, ambos incluidos.

Algoritmo 1: Generador de datos aleatorios

```
for n ← número de alumnos do
    for m ← número de temas do
        respuesta(n, m) ← uniform(−1, 1)
    end
end
```

- Discretización de los datos

  Una vez los datos se encuentran escritos en un fichero, es necesario determinar un umbral que indique cuándo se considera que un alumno es merecedor de un aprobado en un tema en concreto. Esto es, se determina un número comprendido entre -1 y 1 (x) para el cual: Un alumno ha suspendido el tema i-ésimo si su puntuación en la pregunta correspondiente es menor que x y habrá aprobado si su puntuación es mayor o igual que x. En la situación en la que ahora estamos, como la puntuación del alumno es un número real aleatorio comprendido entre -1 y 1, tiene sentido tomar como umbral $x = 0$.

El programa python discretiza entonces los datos que teníamos a ceros y unos, escribiéndolo en un nuevo fichero. Se considera el 0 como un suspenso y el 1 como un aprobado.

Algoritmo 2: Discretización de los datos

```
for n ← número de alumnos do
    for m ← número de temas do
        if respuesta(n, m) < 0 then
            respuesta(n, m) ← 0
        else
            respuesta(n, m) ← 1
        end
    end
end
```

- Detección de prerrequisitos usando probabilidades

  La elección clave de este trabajo es qué procedimiento se sigue para determinar si un tema es prerrequisito de otro o no, a partir de los datos. En cualquier caso, la información que se saca de las grandes cantidades de datos son probabilidades. Podemos calcular la probabilidad de que un determinado alumno apruebe el tema i-ésimo conocida cierta situación con respecto a los otros temas (por ejemplo conociendo que ha aprobado otro tema, o que ha suspendido otros dos . . .). Esto se hace simplemente contando en nuestros datos el número total de casos en los que se da *cierta situación*, y contando de entre ellos cuántos han aprobado el tema i-ésimo. Este cociente nos proporciona la probabilidad buscada.

  La notación usada es la siguiente: Ya sabemos que el 0 va a representar un suspenso y el 1 un aprobado. Por tanto, $P(T_i = 0)$ representa la probabilidad de que un alumno suspenda el tema i-ésimo, sin tener más información. $P(T_i = 1 \mid T_{j_1} = 0, ..., T_{j_n} = 0, T_{k_1} = 1, ..., T_{k_m} = 1)$ representa la probabilidad de que un alumno apruebe el tema i-ésimo, habiendo suspendido los temas $j_1, ..., j_n$ y habiendo aprobado a su vez los temas $k_1, ..., k_m$.

  Así, en nuestro trabajo, hemos considerado la siguiente elección: El tema i-ésimo se considera prerrequisito del tema j-ésimo si y sólo si la probabilidad de que un alumno suspenda el tema j-ésimo habiendo suspendido el tema i-ésimo es mayor que un cierto número θ, dónde θ es un umbral el cuál determinará cómo de fuerte o débil deben ser las dependencias para considerarlas o no prerrequisitos. En resumen, el tema i-ésimo será prerrequisito del tema j-ésimo si y sólo si $P(T_j = 0 \mid T_i = 0) > \theta$. A este valor umbral θ lo vamos a denominar **umbral de fortaleza** y es un número real comprendido entre 0 y 1.

  Esta elección de método tiene sentido intuitivamente pues, si el tema i-ésimo es efectivamente prerrequisito del tema j-ésimo, esto significa que, de alguna manera, necesito los conocimientos del primero para poder manejar correctamente los del segundo. Por tanto, un alumno que tenga un 0 como calificación del tema i-ésimo, tendrá una alta probabilidad de sacar un 0 también en el j-ésimo.

Algoritmo 3: Detección del tema i-ésimo como prerrequisito del tema j-ésimo. Esto es, cálculo e interpretación de $P(T_j = 0 \mid T_i = 0)$

```
total ← 0
concreto ← 0
for n ← número de alumnos do
    if respuesta(n, i) = 0 then
        total ← total + 1
        if respuesta(n, j) = 0 then
            concreto ← concreto + 1
        end
    end
end
p = concreto/total ;                          // p = P(T_j = 0 | T_i = 0)
if p > θ then
    El tema i-ésimo es prerrequisito del tema j-ésimo
else
    El tema i-ésimo no es prerrequisito del tema j-ésimo
end
```

Ejemplo 1. *Veamos en un ejemplo sencillo y con pocos datos, el significado de esta elección de método y su validez. Al igual que en 1.4, vamos a considerar el ejemplo de alumnos cuyo temario a examinar son la operación suma y multiplicación, y queremos saber si son prerrequisitos la una de la otra. Una posible tabla de resultados en la que participan 5 alumnos sería la 2.1; en la que ya entendemos los unos como el correspondiente alumno manejando la operación correspondiente; así como los ceros como desconocimiento de la misma. Vamos a comprobar que la operación suma es prerrequisito de la operación multiplicación, calculando* $P(Mult = 0 \mid Suma = 0)$.

En la tabla hay 2 casos de entre los 5 en los que el alumno no maneja la suma (i.e., $Suma = 0$*), luego el contador* $total = 2$*. De esos casos, ambos cumplen además* $Mult = 0$*, luego* $concreto = 2$ *y obtenemos* $p = P(Mult = 0 \mid Suma = 0) = 2/2 = 1$*. En este caso,* $p = 1 > \theta$ *para todo umbral de fortaleza* $0 \leq \theta < 1$*. Luego para tales valores, la operación suma la consideramos prerrequisito de la multiplicación.*

	Suma	Multiplicación
Alumno 1	1	1
Alumno 2	1	0
Alumno 3	0	0
Alumno 4	1	0
Alumno 5	0	0

Tabla 2.1: Ejemplo de resultados

- Obtención de las tablas de probabilidades condicionadas

  Una vez se conoce qué tema es prerrequisito de cual, se pasa a la siguiente y última fase de cálculo. Ésta consiste en obtener las tablas de probabilidades condicionadas, que es el dato que nos falta para tener en nuestras manos toda la información necesaria para crear una red bayesiana con los resultados. El proceso es un simple

cálculo de probabilidades dadas ciertas situaciones. Para cada tema i-ésimo (que constituye cada nodo de nuestra red), se crea una lista conteniendo los padres del tema en cuestión, esto es, conteniendo los temas que son prerrequisitos del tema. Supongamos que los padres del tema i-ésimo son los temas $j_1, ..., j_n$.

El siguiente paso es considerar todas las posibles situaciones de aprobados/suspensos de los padres. Esto es un trabajo de combinatoria en el que tenemos que tener en cuenta que el tema j_1 puede ser 0 o 1, a su vez el tema j_2 puede ser 0 o 1, y así sucesivamente. Es sencillo ver que en este caso, los n padres del tema i-ésimo, pueden tomar un total de 2^n posibles combinaciones de aprobados/suspensos. Cada una de estas combinaciones constituye una situación. Y se calcula la probabilidad de que un alumno apruebe el tema i-ésimo dadas todas y cada una de las posibles situaciones de los padres. Estas probabilidades (y sus complementarias) son las necesarias para completar nuestras tablas de probabilidades condicionadas.

Algoritmo 4: Cálculo de la tabla de probabilidades condicionadas del tema i-ésimo, cuyos padres son los temas $T_{j_1}, ..., T_{j_n}$

```
for [t_{j_1}, ..., t_{j_n}] ← posibles combinaciones de valores de los padres del tema
 do
    total ← 0
    contador ← 0
    for n ← número de alumnos do
        if respuesta(n,j_i)=t_{j_i} para todo i = 1, ..., n then
            total ← total + 1
            if respuesta(n,i)=1 then
                contador ← contador + 1
            end
        end
    end
    p = contador/total ;          // p = P(T_i = 1 | T_{j_1} = t_{j_1}, ..., T_{j_n} = t_{j_n})
    ;                             // 1-p = P(T_i = 0 | T_{j_1} = t_{j_1}, ..., T_{j_n} = t_{j_n})
end
```

Observación 1. *Si el tema T_i no tiene padres, únicamente necesitamos calcular $P(T_i = 1)$ y su complementario. Con un abuso de notación, incluimos este caso en el algoritmo 4.*

Ejemplo 2. *Seguimos analizando el ejemplo anterior, definido por los datos de la tabla 2.1. Habíamos concluido ya que la operación suma era prerrequisito de la operación multiplicación. Por ello, para construir nuestra red bayesiana necesitamos conocer la probabilidad a priori de la suma (que es un nodo que no tiene padres) y la tabla de probabilidades condicionadas de la multiplicación (cuyo nodo padre es la suma).*

$P(S = 1)$	3/5
$P(S = 0)$	2/5

Tabla 2.2: Probabilidad a priori de la suma

	$S=0$	$S=1$
$P(M=0)$	1	2/3
$P(M=1)$	0	1/3

Tabla 2.3: Tabla de probabilidades condicionadas de la multiplicación

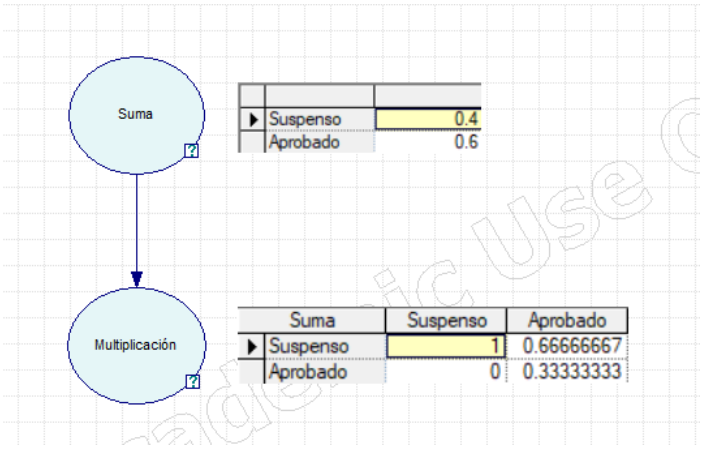


Figura 2.2: Red bayesiana obtenida en el software GeNIe Modeler del Ejemplo 2

- Creación de la red bayesiana visual a través de GeNIe

  Una vez se tiene toda la información sobre la red bayesiana: Los nodos que son los temas de los que tenemos datos, las aristas que representan los prerrequisitos dos a dos y las tablas de probabilidades condicionadas que se acaban de calcular, podemos proceder a crear nuestra red bayesiana visualmente. Gracias a la ayuda del software de GeNIe, los resultados serán fáciles de interpretar. Los datos se trascriben a un fichero que GeNIe es capaz de leer, gracias a la librería *pysmile*, y así se crea la red. Para el ejemplo 2, la interfaz gráfica que nos muestra GeNIe la hemos incluido en la imagen 2.2. Además, ya estamos en condiciones de incluir en nuestra red cualquier tipo de evidencia y obtener resultados. Nuestro primer algoritmo aquí habría acabado.

2.1.2. Errores detectados en el algoritmo y posibles soluciones

El algoritmo definido anteriormente parece funcionar, pero no siempre es así. Al ejecutarlo y al revisar teóricamente el proceso surgen dudas y errores que se deben depurar. Entre ellos encontramos:

- Posible existencia de ciclos en el grafo

  Al definir en el apartado 1.2 las redes bayesianas, quedó claro que éstas debían ser acíclicas. En cambio, ¿Qué nos asegura en este algoritmo la inexistencia de ciclos? Efectivamente, nada. Cuando se ejecuta el algoritmo determinadas veces, creando nuevos datos aleatorios, o cambiando el umbral θ que determina cómo de fuerte ha de ser una dependencia para considerarla prerrequisito, lo más probable es que existan

casos de bucles y por ello GeNIe no pueda construir la red bayesiana correspondiente. Un caso simple sería en el que el tema i-ésimo es prerequiso del tema j-ésimo y viceversa. Esto ocurre numerosas veces en las datos aleatorios pues únicamente se tienen que dar las condiciones $P(T_i = 0 \mid T_j = 0) > \theta$ y $P(T_j = 0 \mid T_i = 0) > \theta$ simultáneamente.

La forma de subsanar los bucles formados por dos temas los cuales resultan ser prerrequisitos entre sí, ha sido juntarlos como si de un único tema se tratase, y se vuelve a aplicar el algoritmo hasta que no existan bucles de dos nodos implicados. La siguiente pregunta a hacerse, y el siguiente problema a resolver será la posible existencia de bucles de más de dos nodos implicados. Vamos a ver, gracias al siguiente teorema, que el problema de existencia de cualquier tipo de bucle se reduce a solucionar el caso sencillo de bucles de dos nodos; de los cuales ya tenemos pensada una posible solución a estudiar en la sección 2.2.

Teorema 1. *Sea $(\mathcal{N}, \mathcal{A})$ un grafo dirigido asociado a una distribución de probabilidad P donde $\mathcal{N}$ es el conjunto de nodos y $\mathcal{A}$ el conjunto de aristas direccionales sobre los nodos tal que $(T_1, T_2) \in \mathcal{A}$ si y sólo si $P(T_2 = 0 \mid T_1 = 0) > \theta$ para θ un umbral de fortaleza determinado. Supongamos que existe un ciclo en el grafo; entonces, el grafo presenta un ciclo de únicamente dos nodos implicados.*

Demostración. Sea pues $(\mathcal{N}, \mathcal{A})$ un grafo dirigido que cumpla las condiciones del teorema. El grafo presenta entonces un ciclo; esto es, existen nodos distintos $T_1, ..., T_n \in \mathcal{N}$ tales que $(T_i, T_{i+1}) \in \mathcal{A}$ para todo $i \in \{1, ..., n-1\}$ además de $(T_n, T_1) \in \mathcal{A}$ para un cierto n natural tal que $n \geq 2$. Esto significa, gracias a la propia definición del grafo y a la distribución de probabilidad asociada al mismo, que $P(T_{i+1} = 0 \mid T_i = 0) > \theta$ para todo $i \in \{1, ..., n-1\}$ además de $P(T_1 = 0 \mid T_n = 0) > \theta$.

Para facilitar la notación vamos a identificar el nodo T_1 indistintamente como T_{n+1} ($T_{n+1} \coloneqq T_1$). Tenemos que probar que existe $i_0 \in \{1, ..., n\}$ tal que $P(T_i = 0 \mid T_{i+1} = 0) > \theta$.

Gracias al teorema de Bayes, sabemos que para todo $i \in \{1, ..., n\}$

$$P(T_i = 0 \mid T_{i+1} = 0) = \frac{P(T_{i+1} = 0 \mid T_i = 0)\, P(T_i = 0)}{P(T_{i+1} = 0)}$$

Pero por hipótesis, $P(T_{i+1} = 0 \mid T_i = 0) > \theta$ para todo $i \in \{1, ..., n\}$. Luego, para todo $i \in \{1, ..., n\}$

$$P(T_i = 0 \mid T_{i+1} = 0) > \theta \, \frac{P(T_i = 0)}{P(T_{i+1} = 0)}$$

Por tanto, existe $i_0 \in \{1, ..., n\}$ tal que $P(T_i = 0 \mid T_{i+1} = 0) > \theta$, que es lo que queremos probar, si y sólo si existe $i_0 \in \{1, ..., n\}$ tal que $P(T_i = 0) \geq P(T_{i+1} = 0)$.

Es evidente pues; por reducción al absurdo, supongamos $P(T_i = 0) < P(T_{i+1} = 0)$ para todo $i_0 \in \{1, ..., n\}$; entonces $P(T_1 = 0) < P(T_2 = 0) < ... < P(T_n = 0)$ y a su vez $P(T_n = 0) < P(T_{n+1} = 0) = P(T_1 = 0)$, luego ya hemos llegado a la contradicción y el teorema queda demostrado.

□

Por tanto, gracias al resultado de este teorema, nos podemos preocupar únicamente por solucionar los ciclos de únicamente dos nodos implicados pues cuando no haya ninguno de éstos, tampoco habrá ningún otro y nuestro grafo será acíclico.

	A	B	C	D
1	1	2	3	4
2	1	0	0	1
3	1	1	0	0
4	0	1	1	1
5	1	1	0	1
6	0	1	1	0
7	1	1	1	1
8	0	1	0	1
9	0	1	0	0
10	0	1	1	1
11	0	0	0	1

Figura 2.3: Tabla de probabilidades condicionadas incompleta pues no encontramos datos donde los temas 2 y 4 estén suspensos simultáneamente, y son prerrequisitos del tema 3

- Posible división por cero

  Otro error encontrado en el algoritmo surge con la propia forma de obtener las tablas de probabilidades condicionadas. Hemos definido las probabilidades a raíz de los datos. Éstas son un cociente $\frac{a}{b}$ dónde b es el número total de veces que se da en los datos una cierta situación y a es el número de veces de entre las primeras en las que se aprueba o se suspende un tema fijo. Pero, no hemos tenido en cuenta la posibilidad de que esa situación no se dé en los datos. Y si esto pasa nos encontraríamos ante un problema de división por cero pues $b = 0$.

  Ejemplo 3. *Mostramos un ejemplo con 10 alumnos y 4 temas en el que tiene lugar esta problemática. Vamos a considerar el valor* $\theta = 0.7$ *como umbral de fortaleza. El problema se encuentra cuando el tema 2 y el tema 4 son ambos prerrequisitos del tema 3. Esto lo podemos ver pues:*

$$P(T_3 = 0 \mid T_2 = 0) = 2/2 = 1 > \theta = 0.7$$
$$P(T_3 = 0 \mid T_4 = 0) = 2/3 > \theta = 0.7$$

  .

  Entonces para rellenar la tabla necesito la probabilidad de aprobar el tema 3, dada una combinación de aprobados/suspensos de los temas 2 y 4. Pero en los datos, no hay ningún caso en el que el tema 2 y el tema 4 estén suspensos a la vez (sean ambos 0), entonces no puedo conocer la probabilidad de aprobar el tema 3 dada esa situación; esto es, no puedo calcular $P(T_3 = 0 \mid T_2 = 0, T_4 = 0)$ *ni su complementario. Porque la situación* $T_2 = 0, T_4 = 0$ *no se da en los datos (véanse los datos en 2.3).*

  Nuestra forma de eliminar dicho problema va a ser suponer que tal probabilidad que no podemos calcular a partir de los datos tiene el valor de 0.5. Al estar el algoritmo diseñado para acabar trabajando con gran cantidad de datos, es como si añadiésemos dos nuevos datos en los que se da la situación, lo cual no afectará notablemente a la conclusión del estudio.

2.2. Algoritmo a prueba de los anteriores errores

2.2.1. Soluciones a los errores encontrados

- Solución de bucles formados por dos nodos

  Supongamos que al aplicar el algoritmo anterior, obtenemos T_i prerrequisito de T_j y viceversa. La solución que se ha adaptado ha sido juntar la información de los dos temas en uno. Para ello, los datos se reescriben y ahora sólo encontramos una columna identificada como $T_{i,j}$ en la que un alumno aparece como aprobado si y sólo si había aprobado ambos temas T_i y T_j. Esto es, la nueva columna correspondiente a $T_{i,j}$ se obtiene realmente como una multiplicación de las columnas que habíamos obtenido anteriormente para T_i y T_j pues tendrá un 1 (un aprobado) en el nuevo tema si y sólo si dicho alumno tenía 1 en ambas columnas antiguas y en cambio suspenderá con que alguno de los temas anteriores tuviera un 0.

  La decisión de esta solución tiene sentido intuitivamente: al encontrarnos dos temas que son prerrequisitos entre sí, nos indica la necesidad de conocer uno para manejar el otro y viceversa; luego resulta claro que deberían ser explicados a la par. Esto es, que podemos considerarlos como un único contenido.

- Solución de inexistencia suficiente de datos para definir todas las probabilidades (división por cero)

  Para resolver este problema, se tiene en cuenta que trabajamos con una gran cantidad de datos. Por ello, el crear un dato ficticio que resuelva la situación no va a modificar de manera significativa el comportamiento del algoritmo en general. Así, la manera de plantear el problema sería la siguiente:

  Supongamos que se ha detectado que los temas $j_1, ...j_n$ son prerrequisitos del tema i-ésimo. Sabemos que la tabla de probabilidades condicionadas correspondientes necesita los datos $P(T_i = t_i \mid T_{j_1} = t_{j_1}, ..., T_{j_n} = t_{j_n})$ donde $t_k \in \{0, 1\}$ para todo $k \in \{i, j_1, ..., j_n\}$. El problema ante el cual nos encontramos es un caso en el que una combinación fija de valores de t_k para $k \in \{j_1, ..., j_n\}$ no se da en los datos y entonces no podemos calcular la correspondiente probabilidad de que un alumno apruebe o suspenda el tema i-ésimo dada dicha combinación de aprobados y suspensos en los temas $j_1, ..., j_n$.

  La propuesta de solución consiste en fijar los valores de tales probabilidades que no podríamos calcular a 0.5. Esto sería lo equivalente a añadir dos nuevos datos ficticios a nuestro fichero con un alumno que apruebe el tema i-ésimo dada la combinación fija de aprobados y suspensos de los temas $j_1, ..., j_n$ y otro que lo suspenda. Como hemos mencionado anteriormente, añadir dos datos cuando manejamos una red inmensa, no va a suponer cambios significativos en el algoritmo. De hecho, para cuantificar la exactitud de los cálculos de las probabilidades condicionadas, se podría contar los casos en los que en los datos encontramos cada una de las combinaciones, así cuanto mayor sea este número, mayor es la precisión con la que hallamos la probabilidad condicionada; esto es, mejor se estará estimando la probabilidad condicionada de la población en general a la cual nuestra muestra representa. En el caso concreto que estamos estudiando, al no haber previamente ningún caso, dicha estimación se va a considerar como "mala". Esto se puede reflejar asignando intervalos de confianza a

cada probabilidad calculada. En este caso obtendríamos una probabilidad de valor 0.5 con intervalo de confianza 0.5; esto es, el valor real de la probabilidad en la población puede tomar cualquier valor entre 0 y 1 realmente (ambos incluidos).

2.2.2. Aplicación del algoritmo mejorado (Ejemplos)

En un primer ejemplo, se simulan las respuestas de 100 alumnos a 5 preguntas, cada una de ellas relacionada con un (y sólo un) tema de la materia.

En la figura 2.4 podemos ver el resultado de una primera ejecución de la primera etapa del programa que se encarga de generar los datos aleatorios (la primera fila corresponde con el título en numeración de cada tema y no con datos aleatorios, por ello los datos aleatorios están recogidos entre las filas 2 y 101). El fichero mostrado en 2.4 se discretiza al nuevo fichero 2.5 como ya explicamos: si el dato es un número $x < 0$ entonces se discretiza como un 0, si en cambio $x \geq 0$ se discretiza como un 1. Así ya tenemos los resultados de los alumnos de forma binaria en aprobados y suspensos.

Ahora, con los datos discretizados en 2.5 como punto de partida, debemos aplicar las siguientes etapas, que son la detección de prerrequisitos usando probabilidades, la obtención de las tablas de probabilidades condicionadas, y por último la creación de la red bayesiana visual a través de GeNIe. La única decisión que se precisa tomar es el valor del umbral de fortaleza, el cual para este ejemplo se ha tomado como $\theta = 0.53$. El resultado obtenido es la red bayesiana mostrada en 2.6. En dicha imagen también se pueden observar las tablas de probabilidades condicionadas obtenidas para cada uno de los nodos de la red gracias a la interfaz de GeNIe.

2.3. Cuantificación de la condición de Prerrequisito

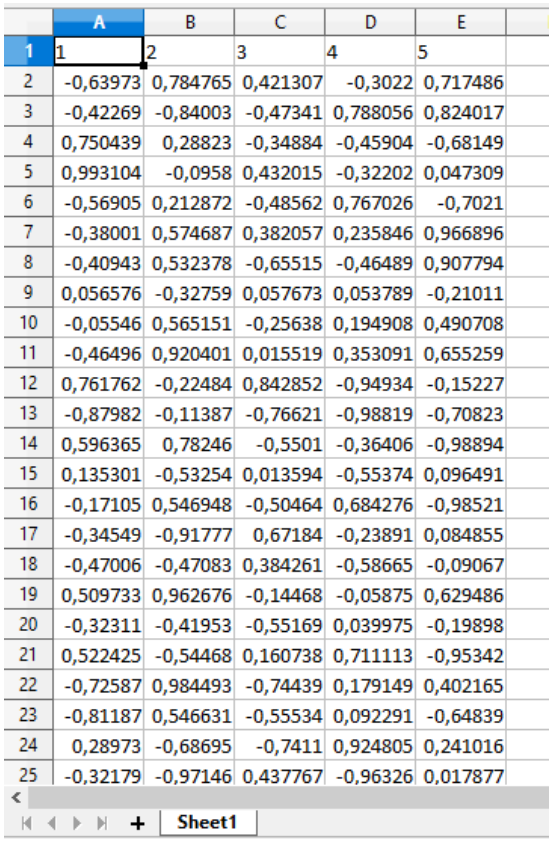
Recordamos que para considerar un tema i-ésimo prerrequisito del j-ésimo tenía que ocurrir que $P(T_j = 0 \mid T_i = 0) > \theta$ dónde θ es el llamado umbral de fortaleza. Parece intuitivo entonces considerar el valor $P(T_j = 0 \mid T_i = 0) \in [0, 1]$ como medida para evaluar cuánto de fuerte es la relación de prerrequisito que los une. Tiene sentido esta elección pues nos indica la probabilidad de que un alumno suspenda el tema j-ésimo, habiendo suspendido ya el i-ésimo. Esta probabilidad es mayor cuánto más se necesita el tema i-ésimo para poder aprender el j-ésimo; esto es, es mayor cuánto más fuerte es la relación de prerrequisitos que los une. Así pues, este valor nos va a dar una vía para poder ordenar los prerrequisitos según sean más o menos fuertes y de esta manera poder establecer un orden óptimo en los contenidos de una materia.

Sea T el conjunto de temas que se analizan en el algoritmo, se define una función $f : T \times T \to [0, 1]$ de la siguiente manera:

$$f(t_1, t_2) \coloneqq P(t_2 = 0 \mid t_1 = 0)$$

. De esta manera, la función f cumple las siguientes propiedades:

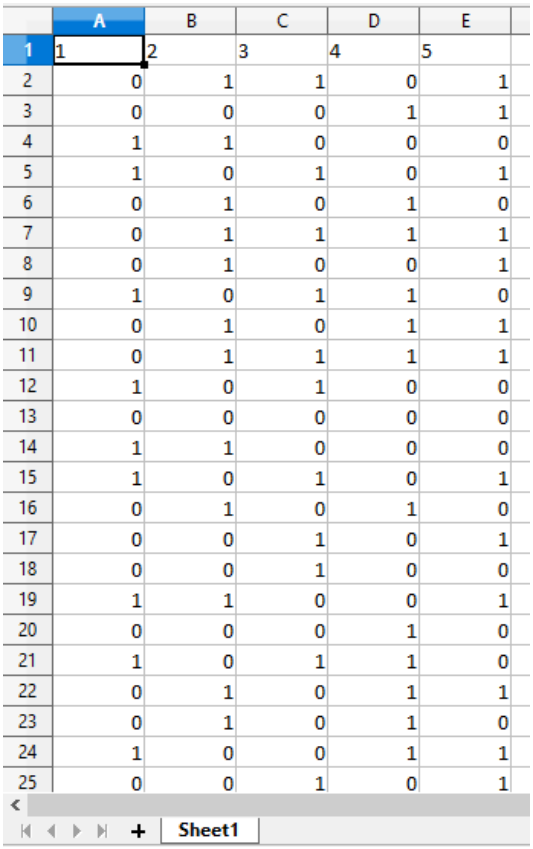
- El tema t_1 es prerrequisito del tema t_2 si y sólo si $f(t_1, t_2) > \theta$ (siendo θ el umbral de fortaleza).



	A	B	C	D	E
1	1	2	3	4	5
2	-0,63973	0,784765	0,421307	-0,3022	0,717486
3	-0,42269	-0,84003	-0,47341	0,788056	0,824017
4	0,750439	0,28823	-0,34884	-0,45904	-0,68149
5	0,993104	-0,0958	0,432015	-0,32202	0,047309
6	-0,56905	0,212872	-0,48562	0,767026	-0,7021
7	-0,38001	0,574687	0,382057	0,235846	0,966896
8	-0,40943	0,532378	-0,65515	-0,46489	0,907794
9	0,056576	-0,32759	0,057673	0,053789	-0,21011
10	-0,05546	0,565151	-0,25638	0,194908	0,490708
11	-0,46496	0,920401	0,015519	0,353091	0,655259
12	0,761762	-0,22484	0,842852	-0,94934	-0,15227
13	-0,87982	-0,11387	-0,76621	-0,98819	-0,70823
14	0,596365	0,78246	-0,5501	-0,36406	-0,98894
15	0,135301	-0,53254	0,013594	-0,55374	0,096491
16	-0,17105	0,546948	-0,50464	0,684276	-0,98521
17	-0,34549	-0,91777	0,67184	-0,23891	0,084855
18	-0,47006	-0,47083	0,384261	-0,58665	-0,09067
19	0,509733	0,962676	-0,14468	-0,05875	0,629486
20	-0,32311	-0,41953	-0,55169	0,039975	-0,19898
21	0,522425	-0,54468	0,160738	0,711113	-0,95342
22	-0,72587	0,984493	-0,74439	0,179149	0,402165
23	-0,81187	0,546631	-0,55534	0,092291	-0,64839
24	0,28973	-0,68695	-0,7411	0,924805	0,241016
25	-0,32179	-0,97146	0,437767	-0,96326	0,017877

Sheet1

Figura 2.4: Fichero excel con los datos aleatorios ya generados



	A	B	C	D	E
1	1	2	3	4	5
2	0	1	1	0	1
3	0	0	0	1	1
4	1	1	0	0	0
5	1	0	1	0	1
6	0	1	0	1	0
7	0	1	1	1	1
8	0	1	0	0	1
9	1	0	1	1	0
10	0	1	0	1	1
11	0	1	1	1	1
12	1	0	1	0	0
13	0	0	0	0	0
14	1	1	0	0	0
15	1	0	1	0	1
16	0	1	0	1	0
17	0	0	1	0	1
18	0	0	1	0	0
19	1	1	0	0	1
20	0	0	0	1	0
21	1	0	1	1	0
22	0	1	0	1	1
23	0	1	0	1	0
24	1	0	0	1	1
25	0	0	1	0	1

Sheet1

Figura 2.5: Fichero excel con los datos aleatorios discretizados a ceros y unos

- La relación "Ser el tema t_1 prerrequisito del tema t_2" es mayor cuánto mayor sea el número $f(t_1, t_2)$.
- La función f no es simétrica, pues generalmente $f(t_1, t_2) \neq f(t_2, t_1)$.
- $f(t_1, t_1) = 1$ para todo $t_1 \in T$

Supongamos ahora que nos encontramos en un caso en el que ambos $f(t_1, t_2) > \theta$ y $f(t_2, t_1) > \theta$ para ciertos $t_1, t_2 \in T$; esto es, nos encontramos ante un caso en el que se formaría un bucle de dos nodos al ser considerado t_1 prerrequisito del tema t_2 y viceversa. Ya hemos explicado en la subsección 2.2.1 que nuestra manera de actuar en este caso sería la unión de ambos temas como un único nodo. Pero, cabe la posibilidad de plantearse una posible detección interna de dependencias de prerrequisitos. Esto va a ser posible gracias a la función definida como f.

Supongamos, sin pérdida de generalidad, que $f(t_1, t_2) \geq f(t_2, t_1)$. Entonces, aún habiendo considerado los temas t_1 y t_2 como un único nodo; internamente, la relación del tema t_1 siendo prerrequisito del tema t_2 es más fuerte que la inversa si la desigualdad es estricta y es igual en el caso de encontrar una igualdad. Por ello, en cualquier caso, si fuera necesario establecer un orden de aprendizaje entre ambos, lo lógico sería que el tema t_1 estuviera en primer lugar.

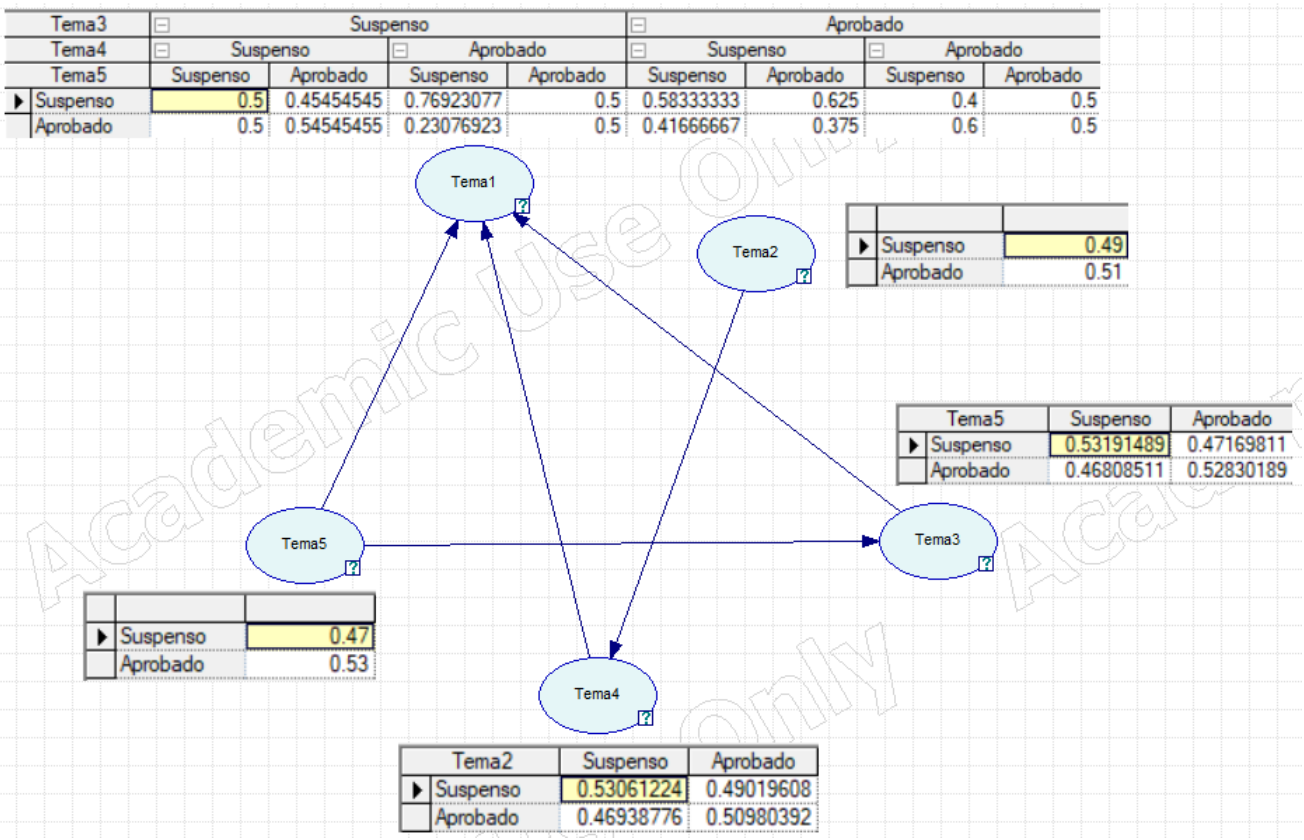


Figura 2.6: Red Bayesiana obtenida en GeNIe al procesar los datos mostrados en 2.5, junto con las tablas de probabilidades condicionadas

Capítulo 3

Aplicación del algoritmo a datos reales

Como ya se ha mencionado, tenemos a mano datos reales de pruebas de evaluación de alumnos obtenidas gracias a la plataforma SIETTE. De la enorme base de datos de dicho sistema, se han escogido los datos correspondientes a la asignatura *Estructuras Discretas* de la UNED, correspondiente al grado de Informática. En concreto, en tales datos, cada alumno contesta a 20 preguntas por sesión, y se le permite realizar varias sesiones. Las preguntas corresponden a cinco temas padre que son: Teoría de Grafos, Teoría de Conjuntos, Funciones, Relaciones y Combinatoria. Y a su vez; cada tema padre se subdivide en temas diferentes. Tenemos como información qué respuesta de cada alumno corresponde a cada subtema. Nuestro objetivo es obtener la estructura de prerrequisitos entre los temas padre de forma automática, para reconocer cuales de ellos son más avanzados y poder crear una secuencia que tenga sentido. Además, vamos a ir más allá, aplicando a su vez el algoritmo a los datos correspondientes de cada tema padre y así poder también obtener la estructura de prerrequisitos entre los subtemas (Esta segunda aplicación habrá que realizarla cinco veces). En este capítulo vamos a analizar los resultados de cada una de las aplicaciones del algoritmo.

Por último, vamos a comparar nuestros resultados con la secuencia de los temas establecida en el sistema SIETTE por el profesor de la asignatura para evaluar si existen posibilidades de mejora.

3.1. Relaciones entre los cinco temas padre

3.1.1. Discretización de los datos reales

Para poder aplicar el algoritmo explicado en el capítulo anterior necesitamos tener los datos ordenados, pero no de cualquier manera. Es necesario que cada fila corresponda a los datos de un alumno y cada columna a los temas de los que nos interesa conocer prerrequisitos. Por ello, ya sabemos que en el caso de analizar relaciones entre los cinco temas padre, hay información repetida. Esto es, un mismo alumno contesta repetidamente a preguntas que corresponden al mismo tema, pudiendo tener diferentes resultados y

calificaciones. Basta con elegir un método que recoja toda esa información y lo simplifique en un 0 o un 1. Existen varias posibilidades como la multiplicación de los resultados, la elección de la última calificación como la usada o, en nuestro caso, la media aritmética de los resultados obtenidos.

El primer paso consiste en escribir los datos ordenados y de manera que sean manejables para poder aplicar el algoritmo. Esto consiste únicamente en una reordenación. Una vez tengamos a nuestra disposición un fichero en el que cada fila i-ésima corresponde a un alumno diferente, cada columna j-ésima a un tema y cada casilla i,j a la media aritmética de las puntuaciones que ha obtenido el alumno correspondiente en todas las preguntas del tema en cuestión; ya estamos en condiciones de aplicar el algoritmo descrito en la sección 2.2 (el último algoritmo descrito con todos los errores solucionados). La primera etapa discretiza cada una de las medias aritméticas encontradas en el fichero a un 0 o un 1 según la media sea superior o inferior a un valor umbral determinado.

Nuestros datos reales, que provienen de tests realizados en SIETTE tienen valores de calificación comprendidos entre -1 y 1. El valor umbral considerado ha sido $x = 0.5$ de manera que si la media supera o iguala el valor, se considera al alumno aprobado y si no, suspenso. Así, con la primera etapa del algoritmo aplicada podemos pasar a la detección de prerrequisitos usando probabilidades.

3.1.2. Estructura de Prerrequisitos entre temas padre

Nos encontramos ahora en condiciones de poder aplicar el algoritmo completo a los datos reales ya discretizados y así analizar resultados. Los datos pasan por todas las etapas explicadas en la sección 2.1.1: detección de prerrequisitos usando probabilidades, obtención de la distribución de probabilidad asociada a la red y la creación de la red bayesiana visual a través de la herramienta GeNIe. Como ya vimos, al proceder a la detección de los prerrequisitos, nos encontramos ante una elección del valor umbral de fortaleza θ tal que el tema i-ésimo es prerrequisito del tema j-ésimo si y sólo si $P(T_j = 0 \mid T_i = 0) > \theta$. Cuanto mayor sea el valor θ, mayor es la dependencia necesaria entre dos temas (esto es, más fuerte ha de ser la relación según la ordenación explicada en 2.3) para considerarlos prerrequisitos. Esto se traduce a su vez en, a mayor valor de θ, menos prerrequisitos entre temas se obtendrán como resultado. Vamos a observar las redes bayesianas obtenidas al aplicar el algoritmo a los datos reales según diferentes valores del parámetro θ.

3.1.3. Secuencia de temas obtenida

A partir de las diferentes redes bayesianas observadas, se puede hacer un análisis de los resultados para producir una secuencia de orden de los cinco temas encontrados. Esto ahorra mucho trabajo al coordinador de la asignatura a la hora de organizar y plantear un posible curso sobre la materia.

En la red 3.1, dada por el umbral de fortaleza $\theta = 0.60$ aparece como único tema sin prerrequisitos el llamado *Teoría de Conjuntos* por lo que debería ser el primer tema en nuestro orden secuencial. A continuación aparece el tema *Relaciones*, que puede colocarse

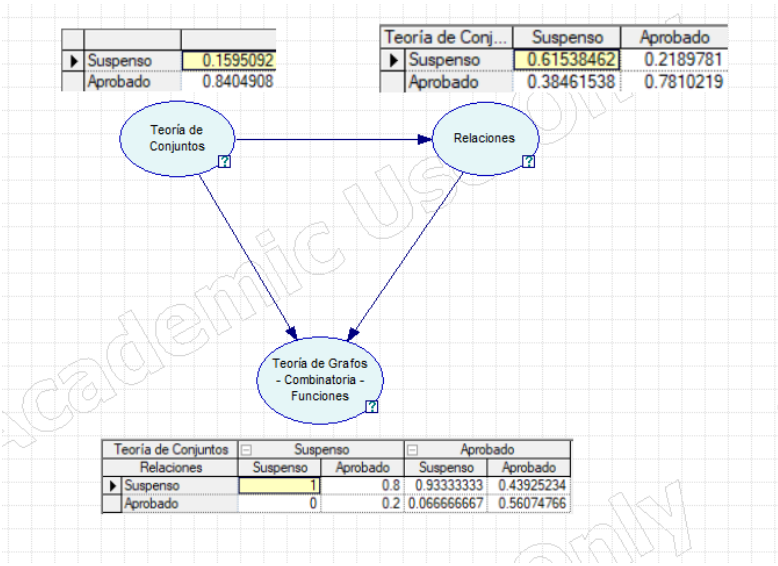


Figura 3.1: Red bayesiana obtenida con valor umbral $\theta = 0.60$

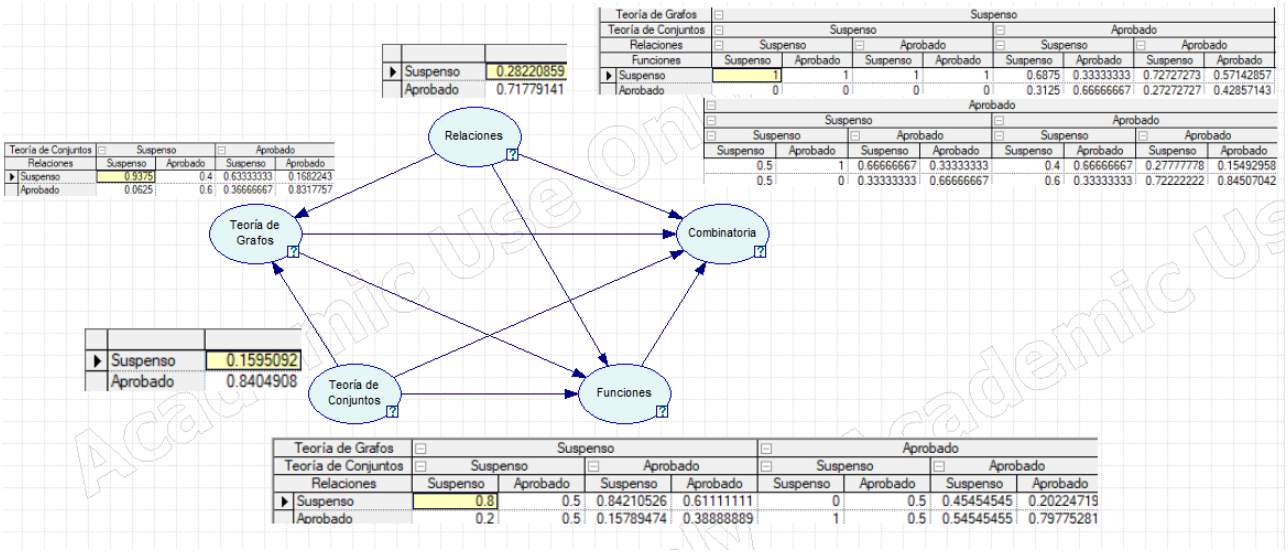


Figura 3.2: Red bayesiana obtenida con valor umbral $\theta = 0.625$

segundo; y por último, observamos cómo los otros tres han sido unidos en uno. Esto es un ejemplo en el que para evitar bucles, los temas automáticamente se identifican como uno único. Para acabar nuestra secuencia entonces, debemos fijarnos en un umbral de fortaleza mayor (pues así encontraremos menos relaciones de prerrequisitos entre los temas) y poder ordenar estos tres temas que nos faltan. El problema acaba cuando nos fijamos en la red dada por el umbral de fortaleza $\theta = 0.625$, que aparece en la imagen 3.2. Aquí encontramos dos nodos sin padres, que corresponden a los temas *Relaciones* y *Teoría de Conjuntos*; por lo cual ambos deberían ser los primeros en nuestra secuencia (sin importar el orden entre ellos), con lo cual no entramos en contradicción con el resultado obtenido previamente. Además, ahora el tema *Teoría de Grafos* tiene como padres a ambos y a ningún otro; por ello es colocado el tercero. Posteriormente, podemos fijar el tema *Funciones* pues sus padres son los tres anteriores y por último y en quinto lugar de nuestra secuencia, debemos colocar a *Combinatoria* pues tiene como prerrequisitos a todos los demás.

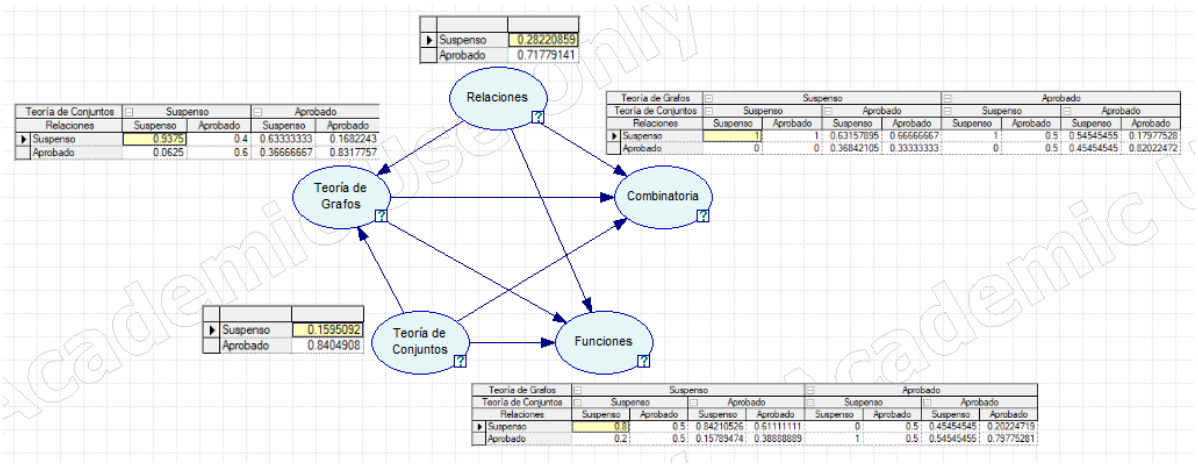


Figura 3.3: Red bayesiana obtenida con valor umbral $\theta = 0.63$

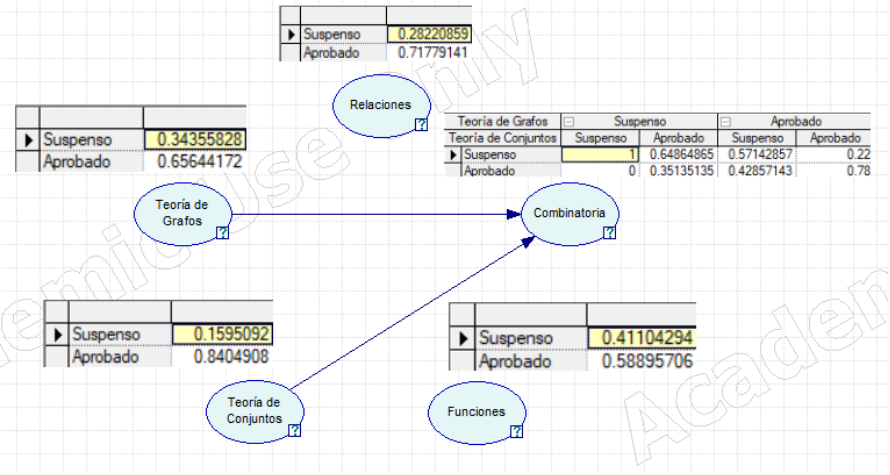


Figura 3.4: Red bayesiana obtenida con valor umbral $\theta = 0.75$

Se presentan las redes bayesianas obtenidas también para los valores $\theta = 0.63$ (3.3) y $\theta = 0.75$ (3.4) para ilustrar cómo en función del valor del parámetro θ y debido a su aumento, el número de relaciones encontradas va disminuyendo y por lo tanto cada vez obtenemos menos información a ser usada para la secuenciación deseada.

En conclusión, y tras el estudio de cada una de las imágenes mostradas en este epígrafe, la secuencia de temas obtenida es la siguiente: **Teoría de Conjuntos - Relaciones - Teoría de Grafos - Funciones - Combinatoria**. Según nuestro estudio, éste sería el orden óptimo de aprendizaje de la materia.

3.2. Relaciones entre los subtemas de cada tema padre

El siguiente paso realizado ha sido aplicar el algoritmo iteradamente, una vez por cada tema padre, pero esta vez aplicado a encontrar las estructuras de prerrequisitos existentes entre los diferentes subtemas de los distintos temas padre. Para ello, los subpasos a realizar

han sido los mismos que en el apartado anterior; comenzando por la reordenación de los datos en cinco ficheros diferentes, uno por cada tema padre a estudiar. En cada uno de ellos encontramos a cada alumno representado en una fila y cada subtema en una columna (la media aritmética de los resultados del alumno en el subtema correspondiente). Posteriormente se han discretizado los datos a ceros y unos de igual manera, tomando como umbral $x = 0.5$. Hasta aquí, la aplicación del algoritmo es idéntica y desearíamos que fuera así pero nos encontramos ante una pequeña dificultad adicional en la aplicación del algoritmo.

3.2.1. Solución para trabajar con casillas vacías

Nos preguntamos si existe la posibilidad de que no tengamos información sobre algún alumno respondiendo a algún subtema en concreto, en cuyo caso nos encontraríamos ante una casilla vacía. Esto no lo hemos tenido en cuenta como posibilidad anteriormente pues únicamente nos interesaban los prerrequisitos entre 5 temas (Teoría de grafos, Teoría de conjuntos, Relaciones, Funciones y Combinatoria) y cada alumno responde 20 preguntas diferentes en cada sesión y suele realizar más de una sesión. Por tanto, siempre tenemos alguna información sobre la calificación del alumno en cada uno de los temas. Sin embargo, con los datos que tenemos de los subtemas, no siempre tiene por qué ser así. De hecho, encontramos numerosos alumnos que no son preguntados por todos los subtemas de todos los temas padre. Por ello, debemos encontrar una solución que permita al algoritmo funcionar a pesar de encontrar casillas vacías de información.

Vamos a trabajar con nuestros ficheros excel de datos incompletos. Para cada fila, se han de realizar ciertos cálculos ya mencionados en el capítulo 2 con el fin de obtener ciertas probabilidades. Al encontrarnos ante una fila en la que necesitemos del valor de una casilla vacía, ignoraremos dicho dato para calcular la probabilidad. Eliminar todos los alumnos con algún subtema vacío no sería una opción pues la mayoría de los mismos se encuentran en tal situación. Sería despreciar demasiada información y nuestro algoritmo habría quedado inválido. Por este motivo, cada una de las filas (que representan a cada una de las colecciones de resultados de un alumno) será ignorada para calcular ciertas probabilidades pero en cambio nos dará información para calcular otras. Esto se traduce en nuestra programación como un simple chequeo antes de calcular las probabilidades. Si las casillas necesarias para determinar una cierta probabilidad que se encuentran en una fila determinada están ocupadas por números; entonces se cuenta con dicha información; en caso contrario, se salta a la fila siguiente.

Cabe mencionar que en esta aplicación del algoritmo, los resultados son mucho menos precisos pues contamos con muchas casillas vacías y mucha menos información de cada subtema. La base de datos es de menor tamaño, por lo que los errores estadísticos aumentan. Esto altera a grandes rasgos la fiabilidad de la secuencia obtenida y de la propia estructura de prerrequisitos que se ha obtenido como fichero de salida en GeNIe.

3.2.2. Estructuras de Prerrequisitos entre subtemas

En las figuras 3.5, 3.6, 3.7, 3.8 y 3.9 se observan las redes bayesianas obtenidas en el software GeNIe al aplicar el algoritmo a cada uno de los datos de los subtemas de cada

tema padre correspondiente. Se ha tomado la decisión de elegir para todos ellos el umbral de fortaleza $\theta = 0.7$ y así tener mismo criterio para todos los casos.

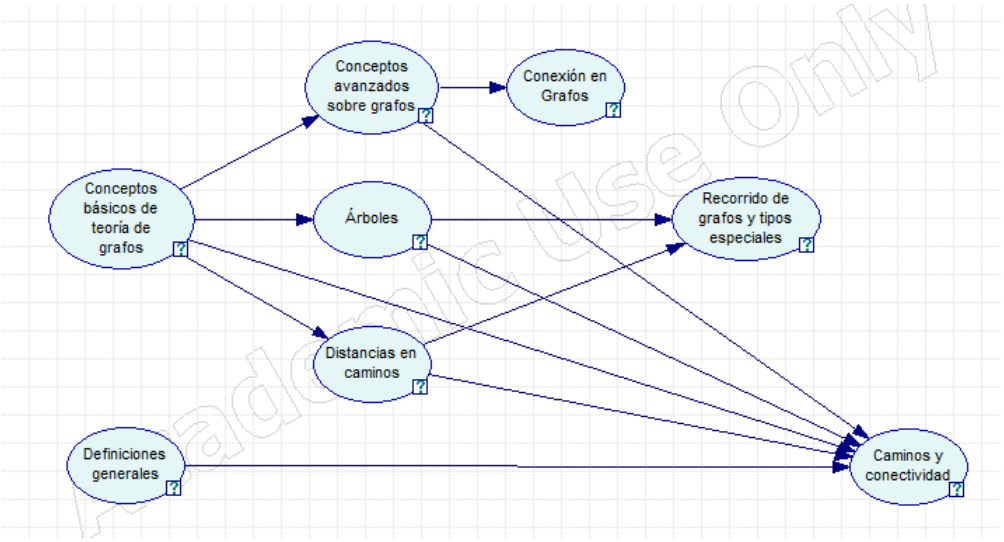


Figura 3.5: Red bayesiana para los subtemas del tema padre: Teoría de Grafos (con $\theta = 0.70$)

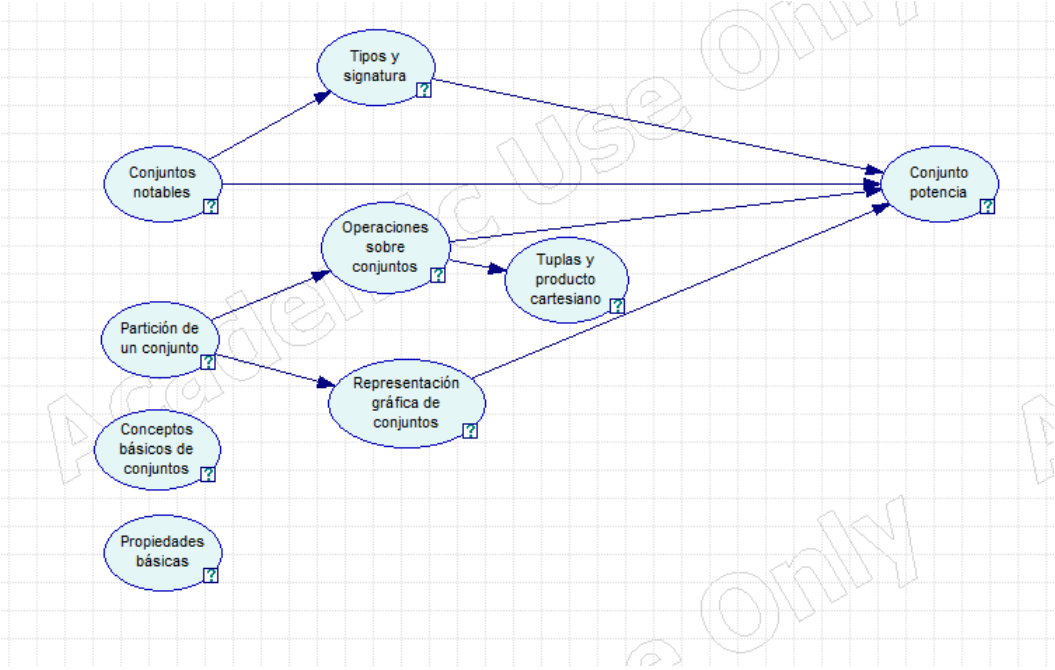


Figura 3.6: Red bayesiana para los subtemas del tema padre: Teoría de Conjuntos (con $\theta = 0.70$)

3.3. Comparación con un modelo de experto

La plataforma SIETTE, cuenta implícitamente con una secuenciación de los temas de cada una de las asignaturas para la selección de preguntas que se realizan a los alumnos. Esta estructura y secuencia, impuesta por el profesor de la correspondiente asignatura, es

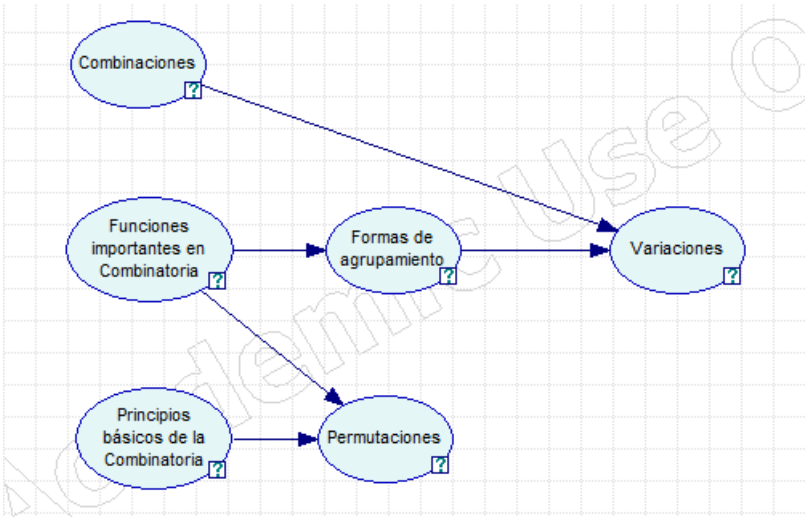


Figura 3.7: Red bayesiana para los subtemas del tema padre: Combinatoria (con $\theta = 0.70$)

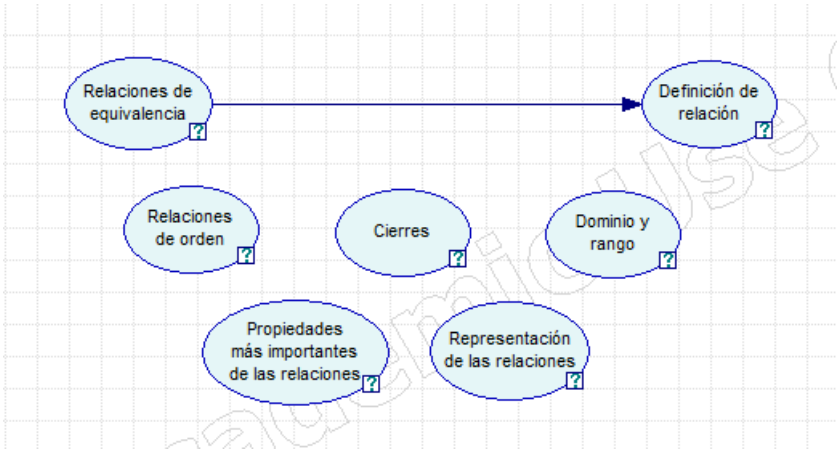


Figura 3.8: Red bayesiana para los subtemas del tema padre: Relaciones (con $\theta = 0.70$)

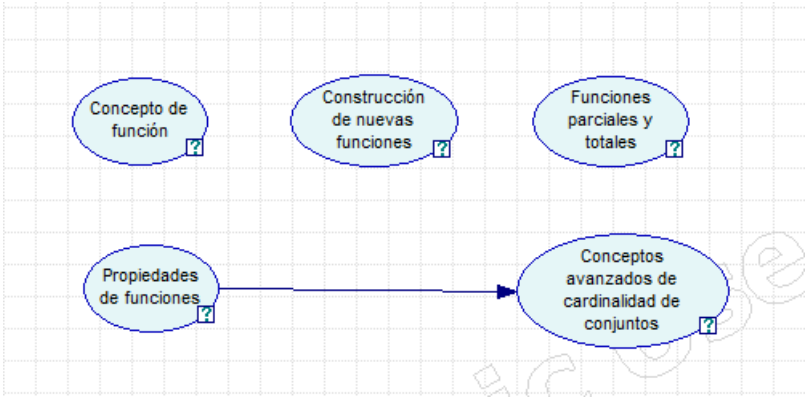


Figura 3.9: Red bayesiana para los subtemas del tema padre: Funciones (con $\theta = 0.70$)

lo que llamamos una estructura modelada por un experto. En este apartado del trabajo vamos a comparar la secuenciación de la asignatura *Estructuras Discretas* que encontramos en la plataforma con la obtenida al aplicar el algoritmo sobre los cinco temas padre.

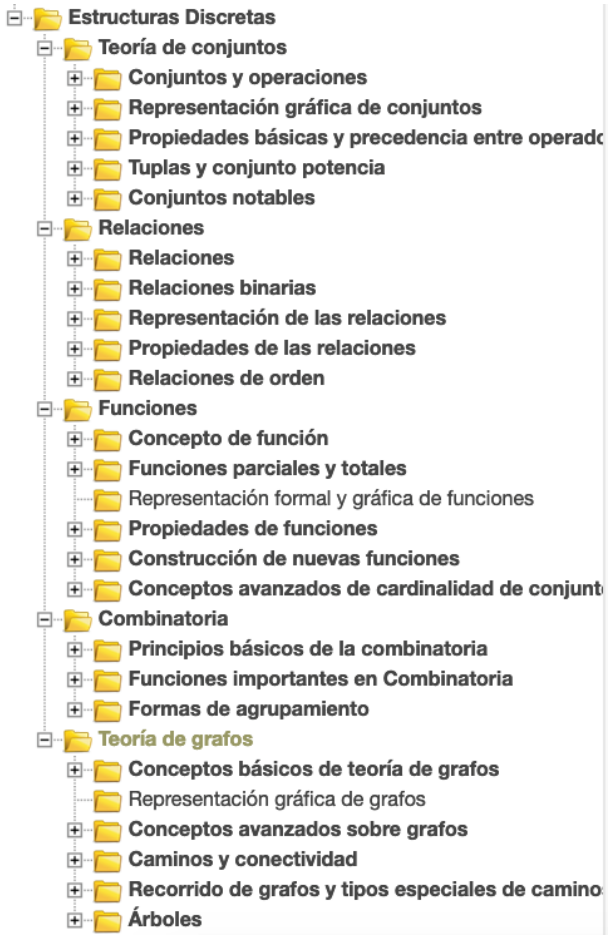


Figura 3.10: Secuenciación de los temas según el modelo de un experto

En la figura 3.10 se observa la secuencia modelada por el profesor de la asignatura al introducir el temario en la plataforma. Los temas están ordenados según aparición; esto es, de arriba a abajo en la figura. La secuencia de temas padre que encontramos entonces es la siguiente: **Teoría de Conjuntos - Relaciones - Funciones - Combinatoria - Teoría de Grafos**. La diferencia con los resultados obtenidos en la sección 3.1.3 que encontramos es el lugar que ocupa el tema *Teoría de Grafos* en la secuencia. Según nuestro algoritmo, había valores de θ para los que éste era prerrequisito de los temas *Funciones* y *Combinatoria*. Por ello, se colocaba en el tercer puesto y recomendábamos su aprendizaje previo a las otras dos unidades de conocimiento. En el modelo del experto en cambio, aparece el último; esto es, es considerado el tema más difícil o el que consta con mayor número de prerrequisitos. El resultado de la comparación nos hace pensar sobre las posibles causas del desacuerdo y propone al profesor un estudio sobre la posibilidad de un cambio en el orden de enseñanza.

La comparación también sería posible para la secuencia de los subtemas pero no nos va a proporcionar un resultado tan claro ni eficaz. Ya se mencionó que en este caso el algoritmo ha trabajado con mucha menos cantidad de datos y los resultados no son muy precisos. Además en el modelo del experto los subtemas no los secuencia; en su lugar, los agrupa en categorías que tengan información común.

Capítulo 4

Conclusiones y líneas de investigación futuras

4.1. Conclusiones

En este último apartado se van a describir las aportaciones más importantes de este trabajo al campo de la aplicación de métodos de inteligencia artificial al estudio de prerrequisitos académicos a partir de resultados de evaluaciones de contenidos a alumnos reales.

Veo necesario destacar lo fundamental que ha sido la formación matemática obtenida durante el estudio del grado para llevar a cabo este trabajo con rigor y precisión. El proceso principal de descubrimiento de prerrequisitos se ha basado en un estudio estadístico, el cual necesitaba de conocimientos en dicha área. También ha sido necesario el rigor propio de las matemáticas a la hora de definir las redes bayesianas y, por supuesto, a la hora de demostrar el teorema 1, que ha sido descubierto según se probaba de forma reiterada el algoritmo con distintos conjuntos de datos. En último lugar, también destaco la adaptación a la escritura de códigos informáticos y al lenguaje *python* que ha requerido este TFG y cómo la formación en programación de un matemático me ha ayudado a llevarlo a cabo de forma satisfactoria.

Ahora sí, en cuanto a los resultados del trabajo, se ha de destacar que el objetivo principal del trabajo ha podido llevarse a cabo de forma satisfactoria; esto es, se ha conseguido obtener una estructura de prerrequisitos en forma de red bayesiana que nos indica las relaciones y dependencias existentes entre los diferentes conceptos de conocimiento. A lo largo del trabajo hemos ido adquiriendo logros parciales, los cuales se van a resumir a lo largo de este apartado.

1. Profundización en el concepto de prerrequisito en un dominio de conocimiento.

   En este TFG se ha estudiado el concepto de *prerrequisito* en un dominio de conocimiento, concretamente asignaturas de enseñanza superior. Se ha provisto a los datos del concepto de *Estructura de Prerrequisitos* y su posible visualización a través de redes bayesianas.

2. Desarrollo de un algoritmo *ad hoc*.

Se ha desarrollado un algoritmo capaz de detectar relaciones de prerrequisitos entre temas y subtemas de una materia cualquiera, y se ha trabajado en una posterior aplicación utilizando datos reales de los resultados parciales de los alumnos en su proceso de aprendizaje. Para ello:

a) Se hizo un estudio del estado del arte del tema, y se revisaron diferentes técnicas y métodos ya existentes, que utilizaron otros autores para resolver este problema. Esto nos ha permitido entender el problema y saber cómo se aborda con otras técnicas de computación, tanto de la Algoritmia como de la Inteligencia Artificial.

b) Se escribió desde cero un algoritmo específico que se ha ido mejorando paulatinamente en un proceso de diseño iterativo. El algoritmo ha evolucionado desde el trabajo con un conjunto de datos muy sencillo (de hecho, de datos generados aleatoriamente según nuestra conveniencia) hasta el uso de datos de usuarios reales cursando una materia de nivel universitario.

3. Implementación del algoritmo en Python.

   Se ha implementado el algoritmo teórico desarrollado en el lenguaje de programación Python para comprobar su funcionamiento y obtener los resultados. Debido a que las relaciones de prerrequisitos son grafos dirigidos, se decidió representar la salida como una red bayesiana ya que simula adecuadamente el razonamiento humano ante la problemática, además de mejorar la eficacia y la exactitud de los resultados [3]. Esta decisión también marcó parte del proceso iterativo ya que hubo que buscar soluciones para evitar ciclos (que son incompatibles con una representación en forma de red bayesiana). Todas las líneas de código se pueden encontrar en el Anexo A.

4. Aplicación a los datos reales.

   En el siguiente paso, se usaron los datos reales procedentes de la plataforma SIETTE como entrada a nuestra implementación del algoritmo en Python. Este paso también participó en el proceso iterativo de corrección del propio método al encontrarnos con una nueva problemática: la posibilidad de trabajar con casillas vacías o el no encontrar ciertas combinaciones de datos necesarias para calcular probabilidades. Se han obtenido en forma de redes bayesianas los resultados, dónde se observan los prerrequisitos encontrados entre temas. Aparecen temas aislados en algunos casos, y temas con muchas relaciones de dependencias en otros.

5. Comprobación de corrección y eficacia del algoritmo.

   Finalmente, una vez descrito el algoritmo, se ha comprobado la eficacia y el correcto funcionamiento de éste gracias al conjunto de datos sobre el que se ha aplicado el algoritmo.

4.1.1. Ventajas de nuestro algoritmo

Según hemos avanzado en las distintas etapas de programación del algoritmo, hemos tomado una serie de decisiones que han hecho que nuestro algoritmo final tenga valor en el campo y sea útil. Algunos de los puntos a destacar son las siguientes:

- El algoritmo es apto para su uso con cualquier conjunto de datos.

  A la hora de programar, se ha hecho hincapié en la creación de un programa adaptable a cualquier conjunto de datos disponible. El único requisito para su uso es la reordenación de los datos previamente en un fichero excel legible por nuestro algoritmo; esto es, que conste de una fila para cada alumno que participe en las pruebas de evaluación y una columna por cada unidad de conocimiento o ítem de entre los que se desean encontrar dependencias; lo cual es tarea sencilla. A partir de ese momento, nuestro algoritmo capta los datos y trabaja con ellos hasta proporcionarnos la red bayesiana en nuestro software visual GeNIe.

- Las dependencias encontradas están cuantificadas.

  En efecto, en el apartado 2.3 vimos que cada relación entre temas encontrada estaba ligada a un número a través de una función f que nos indicaba la fuerza de dicha dependencia. De este modo, se permite una comparación para observar la fuerza de dicha relación e incluso anteponer la información que nos da un prerrequisito considerado fuerte a la hora de crear la secuencia de temas.

- Facilidad de interpretación de los resultados en una secuencia de temas.

  Gracias a la posibilidad de variación del llamado durante el trabajo *umbral de fortaleza* θ, el proceso permite obtener diferentes estructuras de prerrequisitos entre los mismos conceptos de conocimiento. Ya conocemos que a menor valor de θ, mayor número de dependencias encontraremos. De esta forma, si con una única red bayesiana obtenida no queda clara la secuencia de los temas (por existir varias secuencias compatibles con el resultado), podemos modificar el valor de θ e ir consiguiendo más y más información. Este proceso lo hemos aplicado para obtener la secuencia de los cinco temas estudiados en 3.1.3.

4.2. Propuestas y posibles vías de futuro avance

1. Posibilidad de trabajar con rangos de valores.

   Para el desarrollo de nuestro algoritmo, hemos supuesto siempre tener nuestros datos tomando los valores binarios de 0 y 1 como representación de suspensos y aprobados en los temas correspondientes. Si no los teníamos de esta forma, antes de aplicar el algoritmo nos asegurábamos de discretizarlos para que así fuera. Una propuesta para avanzar y mejorar el algoritmo sería poder considerar rangos de valores en vez de únicamente valores discretos (como por ejemplo considerar: Suspenso, Aprobado, Notable y Sobresaliente) o incluso valores numéricos entre 0 y 10 con sus correspondientes decimales. Esto nos propondría mayor información de cada alumno y por ello cabe esperar que con menos cantidad de datos se llegue a la información deseada.

2. Integración del algoritmo en el sistema SIETTE.

   La siguiente propuesta tiene mucho que ver con la comprobación de la eficacia del algoritmo y su posterior aplicación. Se plantea usar el algoritmo con diferentes materias en la plataforma SIETTE y comprobar sus resultados comparándolos con los que cuenta ahora el sistema (que es un modelo de prerrequisitos que hace el profesor

de la asignatura a mano). Lo interesante sería, en un principio, cuando la asignatura se imparte por primera vez, empezar con unas relaciones que señale el profesor (llamado modelo del experto), y una vez que se tenga un conjunto suficientemente grande de datos, generar los prerrequisitos según el proceso de aprendizaje de los propios alumnos, incorporando un mecanismo de retroalimentación. De esta manera generaríamos un sistema de prerrequisitos adaptativo.

3. Comparación del algoritmo con otros métodos.

   En el epígrafe 1.1 vimos diferentes métodos ya existentes para resolver nuestro problema que han usado ya otros autores. Se plantea la idea de implementar los algoritmos de tales propuestas junto al propuesto en este trabajo con el mismo conjunto de datos; pudiendo así comparar los resultados y analizar el por qué de las diferencias encontradas al usar diferentes estrategias. Algunos de los posibles trabajos para una interesante comparación serían: el propuesto en la publicación [7], que usa también redes bayesianas pero con otro enfoque (usa un método denominado COMMAND para seleccionar la red en sí) o cualquier otro trabajo que aplique redes neuronales, otra gran herramienta en auge en el campo de la Inteligencia Artificial.

Apéndice A

Programación en Python

En este apéndice se incluyen todos los programas Python creados para poder llevar a cabo el algoritmo.

A.1. Código para el algoritmo de detección de prerrequisitos en datos aleatorios

Lo primero que hicimos fue la programación del algoritmo para la detección de prerrequisitos de datos aleatorios generados por Python, donde se simulaban respuestas de 100 alumnos diferentes a 5 temas. Se muestra a continuación el código de programación final. Los programas están divididos y ordenados en varios ficheros, donde cada uno de ellos lleva a cabo una etapa del algoritmo mencionada en la subsección del trabajo 2.1.1.

- Generación de datos aleatorios

```
from random import uniform
import xlsxwriter

libro = xlsxwriter.Workbook('100alumnos5temas.xlsx')
hoja = libro.add_worksheet()

numalumnos = 100
numtemas = 5

datos = []
for alumnos in range(numalumnos):
    for temas in range(numtemas):
        datos.append(uniform(-1,1))

for i in range(numtemas):
    hoja.write(0,i,str(i+1))

for i in range(len(datos)):
    hoja.write((i//numtemas)+1, i%numtemas, datos[i])
## El +1 es para insertar la primera fila como los titulos de los
   temas
```

```
21
22 libro.close()
```

- Discretización de los datos

```
1  import xlrd
2  import xlsxwriter
3  librodatos = xlrd.open_workbook('100alumnos5temas.xlsx')
4  hojadatos = librodatos.sheet_by_index(0)
5
6  librodiscretizado = xlsxwriter.Workbook('discretizado1005.xlsx')
7  hojadiscretizada = librodiscretizado.add_worksheet()
8
9  numalumnos = hojadatos.nrows-1
10 numtemas = hojadatos.ncols
11
12 for tema in range(numtemas):
13     hojadiscretizada.write(0,tema,hojadatos.cell_value(0,tema))
14
15 for alumno in range(numalumnos):
16     for tema in range(numtemas):
17         if hojadatos.cell_value(alumno+1,tema)<= 0:
18             hojadiscretizada.write(alumno+1,tema,0)
19         else:
20             hojadiscretizada.write(alumno+1,tema,1)
21
22 librodiscretizado.close()
23 ## LA PRIMERA FILA ES EL IDENTIFICADOR DE CADA TEMA, NO LOS DATOS
      DE UN ALUMNO.
```

- Detección de prerrequisitos usando probabilidades (con la solución a los posibles ciclos dos a dos implementada)

```
1  import xlrd
2  import xlsxwriter
3  from openpyxl import load_workbook
4
5  umbral = 0.7  # Indicamos nuestro umbral de fortaleza
6
7  libro = load_workbook('discretizado1005.xlsx')
8  hoja = libro['Sheet1']
9
10 numalumnos = hoja.max_row-1
11 haybucle = True
12 while (haybucle):
13
14     numtemas = hoja.max_column
15
16     combinaciones = [] # Contiene [[0,1],[1,0],[0,2],...] Así todas
           las pos combinaciones de temas a tener en cuenta.
17     for i in range(numtemas):
18         for j in range(i+1,numtemas):
19             combinaciones.append([i,j])
20             combinaciones.append([j,i])
```

```
21
22      prerrequisitos = []
23
24      #Prerrequisitos va a ser una lista que contiene: Para
           combinacion[i]=[A,B], prerrequisitos[i] contiene [[A,B],[0]]
            si A no es un prerrequisito de B y contiene [[A,B
           ],[1,0.8,0.2]] si A es prerrequisito de B y además si P(B/A)
           =0.8, P(B/noA)=0.2
25      for [t1,t2] in combinaciones:
26          numdeunos = 0
27          num1 = 0.
28          numdeceros = 0
29          num0 = 0.
30          for fila in range(numalumnos):
31              if hoja.cell(row=fila+2,column=t1+1).value==1:
32                  numdeunos+=1
33                  if hoja.cell(row=fila+2,column=t2+1).value==1:
34                      num1+=1
35              else:
36                  numdeceros+=1
37                  if hoja.cell(row=fila+2,column=t2+1).value==0:
38                      num0+=1
39          if (num0/numdeceros)>umbral:
40              prerrequisitos.append([[t1,t2],[1]])
41          else:
42              prerrequisitos.append([[t1,t2],[0]])
43
44      ## Ahora vamos a evitar que A sea prerrequisito de B a la vez
           que B de A, si pasa esto los juntamos como un único tema:
45
46      for i in range(int(len(prerrequisitos)/2)):
47          [[t1,t2],[result1]] = prerrequisitos[2*i]
48          [[t2,t1],[result2]] = prerrequisitos[2*i+1]
49          if (result1==1 and result2==1):
50              acum1 = hoja.cell(row=1,column=t1+1).value
51              acum2 = hoja.cell(row=1,column=t2+1).value
52              hoja.cell(row=1,column=t1+1).value = acum1+'_'+acum2
53              for fila in range(numalumnos):
54                  acum1 = hoja.cell(row=fila+2,column=t1+1).value
55                  acum2 = hoja.cell(row=fila+2,column=t2+1).value
56                  hoja.cell(row=fila+2,column=t1+1).value = acum1*
                       acum2 #CRITERIO: Multiplicación de los datos de
                       los temas t1 y t2
57              hoja.delete_cols(t2+1)
58              break
59      else:
60          haybucle = False
61
62  libro.save('discretizadoreal.xlsx')
```

- Obtención de las tablas de probabilidades condicionadas (con la solución a la posible división por cero implementada)

```
1  from sacoprerrequisitosnobucles import *
2
3  padres = []
```

```
for i in range(numtemas):
    temp = []
    for [[t1,t2],[j]] in prerrequisitos:
        if (t2==i and j==1):
            temp.append(t1)
    padres.append([[i],temp])

print(padres)

## Para sacar la tabla de prob condicionada de cada nodo, vamos a tener que observar 2^n casos si n es el numero de padres que tiene

posibilidades = [] # Aquí sacamos los posibles valores que pueden tomar los padres de cada nodo

for [[t1],padrest1] in padres:
    pos = [[]]
    for padre in padrest1:
        copia = pos.copy()
        for i in range(len(copia)):
            # Aquí ya me aseguro de ordenarlo como me hace falta para luego implementarlo en GeNIe
            ex = pos.pop(0)
            extra = ex.copy()
            ex.append(0)
            extra.append(1)
            pos.append(ex)
            pos.append(extra)
    posibilidades.append([[t1],pos])


## Usando los datos, tomo las prob condicionadas, considerando cada valor que toman los padres

probabilidades = []
for i in range(numtemas):
    [[t1],padrest1] = padres[i]
    [[t1],post1] = posibilidades[i]
    prob = []
    for pos in post1:
        # Para cada posibilidad, cuento las veces que t1=1 de entre todas las veces que ha pasado esa posibilidad
        numpos = 0.
        numtotal = 0.
        for fila in range(numalumnos):
            acum = True
            for j in range(len(padrest1)):
                if hoja.cell(row=fila+2,column=padrest1[j]+1).value!=pos[j]:
                    acum = False
            if acum:
                numtotal += 1
                if hoja.cell(row=fila+2,column=t1+1).value==1:
                    numpos +=1
        if numtotal==0:
            prob.append(0.5) #Aquí arreglo el problema de división por cero. Si no se da el caso, pongo 0.5 en la prob
```

```
54          else:
55              prob.append(numpos/numtotal)
56      probabilidades.append([[t1],prob])
```

- Creación de la red bayesiana visual a través de GeNIe

```
1  import pysmile
2  import pysmile_license
3  from tablasprobcondicionadas import *
4
5  class RedBayesianaGenie:
6
7      def __init__(self):
8          net = pysmile.Network()
9
10         for i in range(numtemas):
11             tema = hoja.cell(row=1,column=i+1).value ##El título
                   del tema que viene en la primera fila del excel
12             self.create_cpt_node(net,"Tema"+tema,"Tema"+tema,["
                   Suspenso","Aprobado"],60+100*(i//2),40+200*(i%2==1))
13
14         for [[t1],padrest1] in padres:
15             tema1 = hoja.cell(row=1,column=t1+1).value
16             for cada in padrest1:
17                 tema2 = hoja.cell(row=1,column=cada+1).value
18                 net.add_arc("Tema"+tema2,"Tema"+tema1);
19
20         for [[t1],prob] in probabilidades:
21             tema1 = hoja.cell(row=1,column=t1+1).value
22             tabla = []
23             for pos in prob:
24                 tabla.append(1-pos)
25                 tabla.append(pos)
26             net.set_node_definition("Tema"+tema1,tabla)
27
28 ## Explicación del orden de estos números: https://support.
       bayesfusion.com/docs/Wrappers/using_arrays.html
29
30         net.write_file("automaticogenie.xdsl")
31
32
33     def create_cpt_node(self, net, id, name, outcomes, x_pos, y_pos
           ):
34         handle = net.add_node(pysmile.NodeType.CPT, id)
35         net.set_node_name(handle, name)
36         net.set_node_position(handle, x_pos, y_pos, 85, 55)
37         initial_outcome_count = net.get_outcome_count(handle)
38
39         for i in range(0, initial_outcome_count):
40             net.set_outcome_id(handle, i, outcomes[i])
41
42         for i in range(initial_outcome_count, len(outcomes)):
43             net.add_outcome(handle, outcomes[i])
44
45         return handle
46
```

```
47
48 ejecuto = RedBayesianaGenie()
```

A.2. Código para la aplicación del algoritmo a los datos reales (temas)

El código mostrado previamente ha sido programado de forma que para todo conjunto de datos que se encuentre ordenada de forma apropiada, sea posible su aplicación. Por ello, con un código de reordenación adaptado a cada conjunto de datos y cambios simples y sencillos en el resto de etapas (como un posible ajuste del umbral de fortaleza, cambio en el nombre de los ficheros de entrada y salida, etc) basta para obtener los resultados deseados.

- Reordenación del conjunto de datos

```
1  import xlrd
2  import xlsxwriter
3  import ast
4  from openpyxl import load_workbook
5
6  # A tener en cuenta: cell_value empieza la cuenta en 0 y cell().
       value empieza la cuenta en 1
7
8  librodatos = xlrd.open_workbook('datosreales.xls')
9  hojadatos = librodatos.sheet_by_index(0)
10
11 librodiscretizado = xlsxwriter.Workbook('ordenadoreal.xlsx')
12 hojadiscretizada = librodiscretizado.add_worksheet()
13 hojaindices = librodiscretizado.add_worksheet()
14
15 librodiscretizado.close()
16
17 librodiscretizado = load_workbook('ordenadoreal.xlsx')
18 hojadiscretizada = librodiscretizado['Sheet1']
19 hojaindices = librodiscretizado['Sheet2']
20
21 filas = hojadatos.nrows-1
22 #(Número de filas con información, quito la 1 que son los títulos)
23
24 numtemas=5
25 hojadiscretizada.cell(1,2).value='18768'
26 hojadiscretizada.cell(1,3).value='24425'
27 hojadiscretizada.cell(1,4).value='24426'
28 hojadiscretizada.cell(1,5).value='24427'
29 hojadiscretizada.cell(1,6).value='20947'
30
31
32
33 for fila in range(1,filas+1):
34     alumno = hojadatos.cell_value(fila,1)
35     tema = hojadatos.cell_value(fila,7)
```

```
36      calif = hojadatos.cell_value(fila,11)
37      numalumnos = hojadiscretizada.max_row-1
38      for i in range(1,numalumnos+1):
39          if hojadiscretizada.cell(i+1,1).value==alumno:
40              for j in range(1,numtemas+1):
41                  if hojadiscretizada.cell(1,j+1).value==str(int(tema
                        )):
42                      puntos = hojadiscretizada.cell(i+1,j+1).value
43                      total = hojaindices.cell(i+1,j+1).value
44                      if puntos is not None:
45                          hojadiscretizada.cell(i+1,j+1).value =
                                puntos+calif
46                          hojaindices.cell(i+1,j+1).value = total+1
47                      else:
48                          hojadiscretizada.cell(i+1,j+1).value =
                                calif
49                          hojaindices.cell(i+1,j+1).value = 1.0
50                      break
51              break
52      else:
53          hojadiscretizada.cell(numalumnos+2,1).value = alumno
54          for j in range(1,numtemas+1):
55              if hojadiscretizada.cell(1,j+1).value==str(int(tema)):
56                  hojadiscretizada.cell(numalumnos+2,j+1).value =
                        float(calif)
57                  hojaindices.cell(numalumnos+2,j+1).value = 1.0
58                  break
59
60
61  ## Ahora ya tenemos cada una de las hojas creadas (sólo falta
        dividir para obtener la media aritmética):
62                  # Hoja1: la suma de todas las calificaciones
63                  # Hoja2: Número total de calificaciones
64
65  nfilas = hojadiscretizada.max_row-1
66  ncolumn = hojadiscretizada.max_column-1
67  for i in range(2,nfilas+2):
68      for j in range(2,ncolumn+2):
69          sumanotas = hojadiscretizada.cell(i,j).value
70          sumatotal = hojaindices.cell(i,j).value
71          hojadiscretizada.cell(i,j).value = sumanotas/sumatotal
72
73
74  librodiscretizado.save('ordenadoreal.xlsx')
```

Una vez ejecutado el código de reordenación se seguiría con los siguientes ficheros ya mostrados con sus correspondientes cambios sencillos a nombres de ficheros y variables: Discretización de los datos, Detección de prerrequisitos usando probabilidades, Obtención de las tablas de probabilidades condicionadas, Creación de la red bayesiana visual a través de GeNIe.

A.3. Código para la aplicación del algoritmo a los datos reales (subtemas)

Cabe esperar que para la segunda aplicación del algoritmo a datos reales, en la que obteníamos las relaciones y dependencias entre subtemas dentro de cada tema padre se necesite un código de reordenación y una posterior aplicación del resto de pasos (sería lo idéntico y equivalente a los pasos seguidos para obtener los prerrequisitos entre temas). Pero en este caso no es así, pues nos encontramos ante la problemática y el trabajo de un conjunto de datos con casillas vacías; y el código necesita una adaptación a esta situación. Realmente, en cada paso en el que usábamos información sobre una casilla fija; es decir, cuando en el código aparece la línea:

```
a = hojadatos.cell_value(alumno,tema)
```

Vamos a comprobar primero que tal línea no está vacía. Si resulta ser no vacía; definimos la variable a y seguimos el código. Si en cambio se da el caso en el que la casilla es vacía, ignoramos tal información como ya explicamos teóricamente. Por tanto, la solución está implementada de la siguiente manera:

```
if type(hojadatos.cell_value(alumno,tema)) is not str :
    a = hojadatos.cell_value(alumno,tema)
```

Además, como queremos que el código se aplique cinco veces idénticas, una para cada tema padre, hemos incluido un bucle *for* al principio de cada una de las etapas que hace esto mismo y nos evita tener que ejecutar por nosotros mismos el código cinco veces. El resultado de todas las etapas con los correspondientes cambios se muestra a continuación:

- Reordenación del conjunto de datos

```
import xlrd
import xlsxwriter
import ast
from openpyxl import load_workbook

librodatos = xlrd.open_workbook('datosreales.xls')
hojadatos = librodatos.sheet_by_index(0)
filas = hojadatos.nrows-1

temaspadre = [18768,24425,24426,24427,20947]
for padres in temaspadre:

    librodiscretizado = xlsxwriter.Workbook('ordenado'+str(padres)+
        '.xlsx')
    hojadiscretizada = librodiscretizado.add_worksheet()
    hojaindices = librodiscretizado.add_worksheet()

    librodiscretizado.close()

```

```
19      librodiscretizado = load_workbook('ordenado'+str(padres)+'.xlsx
            ')
20      hojadiscretizada = librodiscretizado['Sheet1']
21      hojaindices = librodiscretizado['Sheet2']
22
23      for fila in range(1,filas+1):
24          alumno = hojadatos.cell_value(fila,1)
25          temapadre = hojadatos.cell_value(fila,7)
26          tema = hojadatos.cell_value(fila,5)
27          calif = hojadatos.cell_value(fila,11)
28          if temapadre == float(padres):
29              numalumnos = hojadiscretizada.max_row-1
30              numtemas = hojadiscretizada.max_column-1
31              for i in range(1,numalumnos+1):
32                  if hojadiscretizada.cell(i+1,1).value==alumno:
33                      for j in range(1,numtemas+1):
34                          if hojadiscretizada.cell(1,j+1).value==tema
                                :
35                              puntos = hojadiscretizada.cell(i+1,j+1)
                                    .value
36                              total = hojaindices.cell(i+1,j+1).value
37                              if puntos is not None:
38                                  hojadiscretizada.cell(i+1,j+1).
                                        value = puntos+calif
39                                  hojaindices.cell(i+1,j+1).value =
                                        total+1
40                              else:
41                                  hojadiscretizada.cell(i+1,j+1).
                                        value = calif
42                                  hojaindices.cell(i+1,j+1).value =
                                        1.0
43                              break
44                      else:
45                          hojadiscretizada.cell(1,numtemas+2).value =
                                tema
46                          hojadiscretizada.cell(i+1,numtemas+2).value
                                = calif
47                          hojaindices.cell(i+1,numtemas+2).value =
                                1.0
48                      break
49              else:
50                  hojadiscretizada.cell(numalumnos+2,1).value =
                        alumno
51                  for j in range(1,numtemas+1):
52                      if hojadiscretizada.cell(1,j+1).value==tema:
53                          hojadiscretizada.cell(numalumnos+2,j+1).
                                value = float(calif)
54                          hojaindices.cell(numalumnos+2,j+1).value =
                                1.0
55                          break
56                  else:
57                      hojadiscretizada.cell(1,numtemas+2).value =
                            tema
58                      hojadiscretizada.cell(numalumnos+2,numtemas+2).
                            value = calif
59                      hojaindices.cell(numalumnos+2,numtemas+2).value
                            = 1.0
60
```

```
    nfilas = hojadiscretizada.max_row-1
    ncolumn = hojadiscretizada.max_column-1
    for i in range(2,nfilas+2):
        for j in range(2,ncolumn+2):
            sumanotas = hojadiscretizada.cell(i,j).value
            sumatotal = hojaindices.cell(i,j).value
            if sumanotas is not None:
                hojadiscretizada.cell(i,j).value = sumanotas/
                    sumatotal

    librodiscretizado.remove_sheet(hojaindices)
    librodiscretizado.save('ordenado'+str(padres)+'.xlsx')
```

- Discretización de los datos

```
import xlrd
import xlsxwriter

temaspadre = [18768,24425,24426,24427,20947]
for padres in temaspadre:

    librodatos = xlrd.open_workbook('ordenado'+str(padres)+'.xlsx')
    hojadatos = librodatos.sheet_by_index(0)

    librodiscretizado = xlsxwriter.Workbook('discretizado'+str(
        padres)+'.xlsx')
    hojadiscretizada = librodiscretizado.add_worksheet()

    numalumnos = hojadatos.nrows-1
    numtemas = hojadatos.ncols-1

    for tema in range(numtemas):
        hojadiscretizada.write(0,tema+1,hojadatos.cell_value(0,tema
            +1))

    for alumno in range(numalumnos):
        hojadiscretizada.write(alumno+1,0,hojadatos.cell_value(
            alumno+1,0))
        for tema in range(numtemas):
            if type(hojadatos.cell_value(alumno+1,tema+1)) is not
                str :
                if hojadatos.cell_value(alumno+1,tema+1)< 0.5:
                    hojadiscretizada.write(alumno+1,tema+1,0)
                else:
                    hojadiscretizada.write(alumno+1,tema+1,1)

    librodiscretizado.close()
```

- Detección de prerrequisitos usando probabilidades

```
import xlrd
import xlsxwriter
from openpyxl import load_workbook

```

```
umbral = 0.7

prere = []

temaspadre = [18768,24425,24426,24427,20947]
for padres in temaspadre:

    libro = load_workbook('discretizado'+str(padres)+'.xlsx')
    hoja = libro['Sheet1']

    numalumnos = hoja.max_row-1
    haybucle = True
    while (haybucle):

        numtemas = hoja.max_column-1

        combinaciones = []
        for i in range(numtemas):
            for j in range(i+1,numtemas):
                combinaciones.append([i,j])
                combinaciones.append([j,i])

        prerrequisitos = []
        for [t1,t2] in combinaciones:
            numdeceros = 0
            num0 = 0.
            for fila in range(numalumnos):
                if type(hoja.cell(row=fila+2,column=t1+2).value)==
                   int and type(hoja.cell(row=fila+2,column=t2+2).
                   value)==int :
                    if hoja.cell(row=fila+2,column=t1+2).value==0:
                        numdeceros+=1
                        if hoja.cell(row=fila+2,column=t2+2).value
                           ==0:
                            num0+=1

            if numdeceros != 0:
                prob = num0/numdeceros
            else:
                prob = 0.5
            if prob>umbral:
                prerrequisitos.append([[t1,t2],[1]])
            else:
                prerrequisitos.append([[t1,t2],[0]])

        for i in range(int(len(prerrequisitos)/2)):
            [[t1,t2],[result1]] = prerrequisitos[2*i]
            [[t2,t1],[result2]] = prerrequisitos[2*i+1]
            if (result1==1 and result2==1):
                acum1 = hoja.cell(row=1,column=t1+2).value
                acum2 = hoja.cell(row=1,column=t2+2).value
                hoja.cell(row=1,column=t1+2).value = str(acum1)+'_'
                   +str(acum2)
                for fila in range(numalumnos):
                    acum1 = hoja.cell(row=fila+2,column=t1+2).value
                    acum2 = hoja.cell(row=fila+2,column=t2+2).value
                    if type(acum1)==int and type(acum2)==int:
```

```
                            hoja.cell(row=fila+2,column=t1+2).value =
                                acum1*acum2
                        elif type(acum1)==str and type(acum2)==int:
                            hoja.cell(row=fila+2,column=t1+2).value =
                                acum2
                        elif type(acum1)==int and type(acum2)==str:
                            hoja.cell(row=fila+2,column=t1+2).value =
                                acum1
                        elif type(acum1)==str and type(acum2)==str:
                            hoja.cell(row=fila+2,column=t1+2).value =
                                acum1

                    hoja.delete_cols(t2+2)
                    break
            else:
                haybucle = False

            prere.append(prerrequisitos)
        libro.save('discretizado'+str(padres)+'.xlsx')
```

- Obtención de las tablas de probabilidades condicionadas

```
    from csacoprerrequisitossub import *
import xlrd
import xlsxwriter
from openpyxl import load_workbook


probfinal = []
padresfinal = []

temaspadre = [18768,24425,24426,24427,20947]
for indice in range (len(temaspadre)):

    libro = load_workbook('discretizado'+str(temaspadre[indice])+'.
        xlsx')
    hoja = libro['Sheet1']

    numtemas = hoja.max_column-1
    numalumnos = hoja.max_row-1
    padres = []
    prerrequisitos = prere[indice]

    for i in range(numtemas):
        temp = []
        for [[t1,t2],[j]] in prerrequisitos:
            if (t2==i and j==1):
                temp.append(t1)
        padres.append([[i],temp])
    padresfinal.append(padres)


    posibilidades = []
    for [[t1],padrest1] in padres:
        pos = [[]]
        for padre in padrest1:
```

```
34              copia = pos.copy()
35              for i in range(len(copia)):
36                  ex = pos.pop(0)
37                  extra = ex.copy()
38                  ex.append(0)
39                  extra.append(1)
40                  pos.append(ex)
41                  pos.append(extra)
42          posibilidades.append([[t1],pos])
43
44      probabilidades = []
45      for i in range(numtemas):
46          [[t1],padrest1] = padres[i]
47          [[t1],post1] = posibilidades[i]
48          prob = []
49          for pos in post1:
50              numpos = 0.
51              numtotal = 0.
52              for fila in range(numalumnos):
53                  acum = True
54                  for j in range(len(padrest1)):
55                      if hoja.cell(row=fila+2,column=padrest1[j]+2).
                           value!=pos[j]:
56                          acum = False
57                  if acum and type(hoja.cell(row=fila+2,column=t1+2).
                       value)==int:
58                      numtotal += 1
59                      if hoja.cell(row=fila+2,column=t1+2).value==1:
60                          numpos +=1
61              if numtotal==0:
62                  prob.append(0.5)
63              else:
64                  prob.append(numpos/numtotal)
65          probabilidades.append([[t1],prob])
66      print(probabilidades)
67
68      probfinal.append(probabilidades)
```

- Creación de la red bayesiana visual a través de GeNIe

```
1       import pysmile
2   import pysmile_license
3   from dtablasprobcondsub import *
4
5   temaspadre = [18768,24425,24426,24427,20947]
6   for indice in range (len(temaspadre)):
7
8       libro = load_workbook('discretizado'+str(temaspadre[indice])+'.
            xlsx')
9       hoja = libro['Sheet1']
10
11      numtemas = hoja.max_column-1
12      prerrequisitos = prere[indice]
13      probabilidades = probfinal[indice]
14      padres = padresfinal[indice]
15
```

```
16      class RedBayesianaGenie:
17
18          def __init__(self):
19              net = pysmile.Network()
20
21              for i in range(numtemas):
22                  tema = hoja.cell(row=1,column=i+2).value
23                  self.create_cpt_node(net,"Tema"+str(tema),"Tema"+
                        str(tema),["Suspenso","Aprobado"],60+100*(i//2)
                        ,40+200*(i%2==1))
24
25              for [[t1],padrest1] in padres:
26                  tema1 = hoja.cell(row=1,column=t1+2).value
27                  for cada in padrest1:
28                      tema2 = hoja.cell(row=1,column=cada+2).value
29                      net.add_arc("Tema"+str(tema2),"Tema"+str(tema1)
                            );
30
31              for [[t1],prob] in probabilidades:
32                  tema1 = hoja.cell(row=1,column=t1+2).value
33                  tabla = []
34                  for pos in prob:
35                      tabla.append(1-pos)
36                      tabla.append(pos)
37                  net.set_node_definition("Tema"+str(tema1),tabla)
38
39              net.write_file(str(temaspadre[indice])+".xdsl")
40
41
42          def create_cpt_node(self, net, id, name, outcomes, x_pos,
                y_pos):
43              handle = net.add_node(pysmile.NodeType.CPT, id)
44              net.set_node_name(handle, name)
45              net.set_node_position(handle, x_pos, y_pos, 85, 55)
46              initial_outcome_count = net.get_outcome_count(handle)
47
48              for i in range(0, initial_outcome_count):
49                  net.set_outcome_id(handle, i, outcomes[i])
50
51              for i in range(initial_outcome_count, len(outcomes)):
52                  net.add_outcome(handle, outcomes[i])
53
54              return handle
55
56
57      ejecuto = RedBayesianaGenie()
```

Bibliografía

[1] Michael T. Brannick. Item response theory. url: `http://luna.cas.usf.edu/~mbrannick/files/pmet/irt.htm`.

[2] Emma Brunskill. Estimating prerequisite structure from noisy data. In *EDM*, 2011.

[3] C. Carmona, E. Millán, J. L. Pérez-de-la Cruz, M. Trella, and R. Conejo. Introducing prerequisite relations in a multi-layered bayesian student model. In Liliana Ardissono, Paul Brna, and Antonija Mitrovic, editors, *User Modeling 2005*, pages 347–356, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[4] Devendra Chaplot, Yiming Yang, Jaime Carbonell, and Kenneth Koedinger. Data-driven automated induction of prerequisite structure graphs. 06 2016. `https://educationaldatamining.org/EDM2016dproceedings/paper_149.pdf`.

[5] Weiyu Chen, Andrew S. Lan, Da Cao, Christopher G. Brinton, and Mung Chiang. Behavioral analysis at scale: Learning course prerequisite structures from learner clickstreams. In Kristy Elizabeth Boyer and Michael Yudelson, editors, *Proceedings of the 11th International Conference on Educational Data Mining, EDM 2018, Buffalo, NY, USA, July 15-18, 2018*. International Educational Data Mining Society (IEDMS), 2018.

[6] Yang Chen, Pierre-Henri Wuillemin, and Jean-Marc Labat. Discovering prerequisite structure of skills through probabilistic association rules mining. In Olga C. Santos, Jesus Boticario, Cristóbal Romero, Mykola Pechenizkiy, Agathe Merceron, Piotr Mitros, José María Luna, Marian Cristian Mihaescu, Pablo Moreno, Arnon Hershkovitz, Sebastián Ventura, and Michel C. Desmarais, editors, *Proceedings of the 8th International Conference on Educational Data Mining, EDM 2015, Madrid, Spain, June 26-29, 2015*, pages 117–124. International Educational Data Mining Society (IEDMS), 2015.

[7] Yetian Chen, José P. González-Brenes, and Jin Tian. Joint discovery of skill prerequisite graphs and student models. In *The 9th International Conference on Educational Data Mining*, North Carolina, USA, 2016.

[8] Ricardo Conejo, Eduardo Guzmán, Eva Millán, Mónica Trella, José-Luis Pérez-de-la Cruz, and Antonia Ríos. Siette: a web-based tool for adaptive testing. *I. J. Artificial Intelligence in Education*, 14:29–61, 01 2004.

[9] Díez F.J. *Introducción al Razonamiento Aproximado*. Dpto. Inteligencia Artificial, UNED, noviembre 2005 edition, 1998.

[10] I. Goldin, R. Scheines, and E. Silver. Discovering prerequisite relationships among knowledge components. In *Proc. International Conference on Educational Data Mining*, pages 355–356, July 2014.

[11] Soo-Yun Han, Jiyoung Yoon, and Yun Joo Yoo. Discovering skill prerequisite structure through bayesian estimation and nested model comparison. In *EDM*, 2017. `https://pdfs.semanticscholar.org/b534/4b10602ab6f17044aab208ba4ef9004d783a.pdf`.

[12] Igor Labutov, Yun Huang, Peter Brusilovsky, and Daqing He. Semi-supervised techniques for mining learning outcomes and prerequisites. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, page 907–915, New York, NY, USA, 2017. Association for Computing Machinery.

[13] Andrew S. Lan, Christoph Studer, and Richard G. Baraniuk. Time-varying learning and content analytics via sparse factor analysis. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, page 452–461, New York, NY, USA, 2014. Association for Computing Machinery.

[14] Chen Liang, Zhaohui Wu, Wenyi Huang, and C. Lee Giles. Measuring prerequisite relations among concepts. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1668–1674, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

[15] Siddharth Reddy, Igor Labutov, and Thorsten Joachims. Learning student and content embeddings for personalized lesson sequence recommendation. In *Proceedings of the Third (2016) ACM Conference on Learning @ Scale*, L@S '16, page 93–96, New York, NY, USA, 2016. Association for Computing Machinery.

[16] Ana Álvarez Suárez. Predicción de separaciones en aeronaves mediante redes bayesianas. Master's thesis, E.T.S. de Ingenieros Informáticos (UPM)., Madrid, España, 2017.