CLASIFICADORES POR REDES BAYESIANAS

por

Carlos López de Castilla Vásquez

Tesis sometida en cumplimento parcial de los requisitos para el grado de

MAESTRO EN CIENCIAS
en
MATEMATICA
(Estadística)

UNIVERSIDAD DE PUERTO RICO
MAYAGÜEZ CAMPUS
2005

Aprobada por:

_____	_____
Julio Quintana, PhD Miembro, Comité Graduado	Fecha
_____	_____
Dámaris Santana, PhD Miembro, Comité Graduado	Fecha
_____	_____
Edgar Acuña, PhD Presidente, Comité Graduado	Fecha
_____	_____
José R. Cedeño, PhD Representante de Estudios Graduados	Fecha
_____	_____
Pedro Vasquez, DSc Director de Departamento	Fecha

ABSTRACT

A Bayesian network is a compact representation of joint probability function. Formally, a Bayesian network is an acyclic directed graph in which each node represents a random variable and the relationships of dependence and independence are established by the structure of the network. In this thesis the performance of some classifiers constructed in basis of estimation of Bayesian networks is evaluated. The classifiers considered are Naïve Bayes, TAN and Bayesian multinets. The evaluation is carried out on 20 datasets for the UCI Machine Learning Repository. Our results show that classifiers based on the structure of a Bayesian network have a good predictive performance.

ABSTRACTO

Una red Bayesiana es una representación compacta de una función de probabilidad conjunta. Formalmente, una red Bayesiana es un grafo acíclico dirigido en el que cada nodo representa una variable aleatoria y las relaciones de dependencias e independencias condicionales quedan establecidas en la propia estructura de la red. En este trabajo se evalúa el comportamiento de algunos clasificadores construidos en base a la teoría de estimación por redes Bayesianas. Se consideran los clasificadores Naive Bayes, TAN y las multiredes Bayesianas CL. Estos clasificadores se comparan usando 20 conjuntos de datos provenientes del repositorio UCI Machine Learning. Nuestros resultados muestran que los clasificadores basados en la estructura de una red Bayesiana presentan un buen comportamiento predictivo.

Para mi familia y mi esposa Melissa

AGRADECIMIENTOS

Al doctor Edgar Acuña por su interés, apoyo y oportunas sugerencias al presente documento.

Tabla de Contenidos

Lista de Tablas

Tablas **Página**

Lista de Figuras

Figuras **Página**

1 INTRODUCCION

Una red Bayesiana es un modelo gráfico que permite representar las relaciones de dependencia entre un conjunto de variables. Las redes Bayesianas se usan para codificar el conocimiento en los sistemas expertos [21] y existen numerosos algoritmos desarrollados para estimar este tipo de redes. Las redes Bayesianas, junto con los métodos Bayesianos, facilitan la combinación del conocimiento experto con la información contenida en un conjunto de datos.

Una red Bayesiana es un grafo acíclico dirigido en el que cada nodo representa una variable aleatoria que tiene asociada una función de probabilidad condicional. La estructura de la red Bayesiana provee información sobre las relaciones de dependencia e independencia condicional existentes entre las variables. Estas relaciones simplifican la representación de la función de probabilidad conjunta como el producto de las funciones de probabilidad condicional de cada variable.

El proceso de estimación de una red Bayesiana consiste de una etapa de aprendizaje estructural y una etapa de aprendizaje paramétrico [24]. La primera etapa consiste en obtener la estructura de la red y la segunda tiene por finalidad estimar los parámetros de las funciones de probabilidad condicional. La estructura puede especificarse por alguien con cierto conocimiento, de ahí que un nombre para las redes Bayesianas es también el de sistemas probabilísticos expertos; un sistema experto por que la red codifica en su

estructura el conocimiento experto de un problema y probabilístico por que las dependencias entre las variables así lo son.

1.1 Motivación

La estructura de una red Bayesiana puede utilizarse para construir clasificadores identificando uno de los nodos con la variable de clase. El proceso de clasificación se realiza usando el teorema de Bayes para asignar una observación en la clase con mayor probabilidad. Sin embargo, el procedimiento anterior requiere el uso de la función de probabilidad conjunta de las variables involucradas. Una red Bayesiana permite representar esta función de probabilidad conjunta usando funciones de probabilidad definidas en menos variables. Esto permite simplificar el cálculo de las probabilidades de asignación.

1.2 Objetivos

- Estimar la estructura de una red Bayesiana a partir del conjunto de datos de tal forma que queden establecidas, de manera clara, las relaciones de dependencia e independencia condicional entre las variables.
- Construir los clasificadores Naive Bayes, TAN y las multiredes Bayesianas CL en base a la estructura de la red, usando un conjunto de datos para estimar las probabilidades condicionales.
- Comparar la precisión de cada clasificador usando conjuntos de datos de UCI Machine Learning.

1.3 Resumen de los Capítulos

La presente tesis esta estructurada en seis capítulos. El capítulo 2 brinda algunos conceptos básicos relacionados con la teoría de las redes Bayesianas. En el capítulo 3 se definen de manera formal estas redes y se mencionan algunos métodos de estimación de la estructura y los parámetros. El capítulo 4 presenta algunos clasificadores construidos en base a la estructura de las redes Bayesianas tales como Naive Bayes, TAN y las multiredes Bayesianas CL. El capítulo 5 muestra los resultados obtenidos usando los clasificadores anteriores sobre 20 conjuntos de datos provenientes de UCI Machine Learning. Finamente, el capítulo 6 presenta las conclusiones obtenidas y menciona algunos trabajos de investigación sobre el proceso de clasificación usando redes Bayesianas con variables discretas y contínuas.

2 ASPECTOS TEÓRICOS

2.1 Introducción

Un modelo gráfico puede usarse para representar las relaciones de dependencia entre un conjunto de variables. Los nodos corresponden a las variables aleatorias y las relaciones de dependencia existentes pueden representarse usando arcos entre los nodos. La interpretación gráfica de estas relaciones de dependencia en los modelos gráficos constituye una de sus características principales.

Las redes Bayesianas se encuentran entre los modelos gráficos más populares. La principal diferencia, con respecto a otros modelos, está en que sus arcos son dirigidos y representan dependencia condicional entre las variables. El nombre proviene del hecho que gran parte de la teoría relevante con este tipo de redes se basa en la estadística Bayesiana. En esta sección se presentan algunos temas relacionados con el uso y la estimación de las redes Bayesianas.

2.2 Teorema de Bayes

En su forma básica, el teorema de Bayes es un simple resultado de probabilidades condicionales. Sean A y B dos eventos con $p(\mathrm{A}) > 0$. Entonces

$$p(\mathrm{B}/\mathrm{A}) = \frac{p(\mathrm{B})\, p(\mathrm{A}/\mathrm{B})}{p(\mathrm{A})} \tag{2.1}$$

El uso principal de este teorema, en aplicaciones de probabilidad, es revertir el condicionamiento de los eventos, esto es, mostrar cómo la probabilidad de B/A está relacionada con la de A/B.

2.3 La regla de la cadena

La función de probabilidad conjunta para las variables aleatorias $X_1, \cdots, X_n$ puede expresarse como el producto de funciones condicionadas de probabilidad

$$p(X_1, \cdots, X_n) = p(X_1) \prod_{i=2}^{n} p(X_i / X_1, \cdots, X_{i-1}) \tag{2.2}$$

2.4 Independencia condicional

Dos variables aleatorias discretas X e Y se dice que son independientes si $p(X) = p(X/Y)$. Esto puede expresarse como $p(X, Y) = p(X)\, p(Y)$. En el caso de tres variables aleatorias discretas X, Y y Z se dice que X e Y son condicionalmente independientes del valor de Z si la distribución de probabilidad de X es independiente del valor de Y dado el valor de Z, esto es $p(X/Y, Z) = p(X/Z)$. Extendiendo esta definición para un conjunto de variables puede expresarse que $X_1, \cdots, X_n$ es condicionalmente independiente de $Y_1, \cdots, Y_m$ dado el conjunto de variables $Z_1, \cdots, Z_p$ si

$$p\left(X_1,\cdots,X_n/Y_1,\cdots,Y_m,Z_1,\cdots,Z_p\right)=p\left(X_1,\cdots,X_n/Z_1,\cdots,Z_p\right) \quad (2.3)$$

2.5 Entropía

La entropía de la variable aleatoria X se define por $\mathrm{H}_p(X)=-\sum_X p(X)\log p(X)$, donde p es la función de probabilidad de X. La función $\mathrm{H}_p(X)$ es el número óptimo de bits necesarios para almacenar los valores de X, en otras palabras mide la cantidad de información contenida en X. La entropía es siempre no negativa. La entropía es cero si y sólo si X es perfectamente predecible, es decir si un valor de X tiene probabilidad uno. La entropía es máxima cuando $p(X)$ es totalmente no informativa, es decir cuando se asigna una probabilidad uniforme para X. En este caso $\mathrm{H}_p(X)=\log\|X\|$, donde $\|X\|$ es el número de valores diferentes que puede tomar X.

La entropía condicional $\mathrm{H}_p(X/Y)=-\sum_Y p(Y)\sum_X p(X/Y)\log p(X/Y)$ mide la entropía de X cuando se conoce el valor de Y. En términos de codificación, $\mathrm{H}_p(X/Y)$ mide el número óptimo de bits necesarios para codificar el valor de X cuando se conoce el valor de Y. Conocer el valor de Y puede ser útil para codificar X de forma más compacta. De hecho, $\mathrm{H}_p(X/Y)\le\mathrm{H}_p(X)$. La diferencia entre estos dos valores, $\mathrm{I}_p(X;Y)=\mathrm{H}_p(X)-\mathrm{H}_p(X/Y)$, es llamada la información mutua entre X e Y que mide cuanta información Y refiere sobre X.

2.6 Estadística Bayesiana

La estadística Bayesiana le debe su nombre al trabajo pionero del reverendo Thomas Bayes titulado "*An Essay towards solving a Problem in the Doctrine of Chances*" publicado póstumamente en 1764 en la "*Philosophical Transactions of the Royal Society of London*". Aunque la obra de Thomas Bayes data ya de hace más de dos siglos, la estadística Bayesiana es relativamente nueva, y actualmente ostenta un gran desarrollo aunque no ajeno también a grandes controversias.

El marco teórico en el cual se desarrolla la inferencia Bayesiana es idéntico al de la teoría clásica. Se tiene un parámetro poblacional θ sobre el cual se desea hacer inferencia y se tiene un modelo de probabilidad $p(\mathbf{X}/\theta)$. La diferencia fundamental entre la teoría clásica y la Bayesiana está en que θ es tratado como una cantidad aleatoria. Así, la inferencia Bayesiana se basa en la función de probabilidad posterior, $p(\theta/\mathbf{X})$, en lugar de $p(\mathbf{X}/\theta)$, esto es, en la función de probabilidades del parámetro dados los valores de los datos. Además, es necesario especificar una función de probabilidades a priori $p(\theta)$, la cual representa el conocimiento que se tiene sobre la función de probabilidades de θ previo a la obtención de los datos.

Esta noción de una función de probabilidad a priori para el parámetro constituye el centro del pensamiento Bayesiano y, dependiendo de si se es un defensor o un opositor a esta

metodología, su principal ventaja sobre la teoría clásica o su mayor vulnerabilidad.

Un problema de la estadística Bayesiana es que, a menos que la función de probabilidad a priori sea seleccionada cuidadosamente, la función de probabilidad posterior resultante puede no pertenecer a ninguna familia conocida y el tratamiento matemático consiguiente podría complicarse notablemente. Para resolver este problema, una posibilidad consiste en elegir la función de probabilidad a priori entre la clase de distribuciones conjugadas. Si las funciones de probabilidad a priori y posterior pertenecen a la misma familia de distribuciones para un mismo proceso muestral, se dice que son conjugadas naturales con respecto al mismo.

Desde la perspectiva Bayesiana el estimador de máxima verosimilitud para un parámetro θ es la moda más grande de la función de probabilidad posterior $p(\theta/\mathbf{X})$. Sin embargo la media y la mediana pueden ser también buenos estimadores de θ y su cálculo a través de una distribución conocida para $p(\theta/\mathbf{X})$ resulta, por lo general, sencillo.

2.7 Grafos

Un modelo probabilístico puede definirse usando un grafo que describa las relaciones existentes entre las variables. Supóngase un conjunto de variables $\mathbf{X} = \{X_1, \cdots, X_n\}$ que pueden relacionarse entre sí. El conjunto anterior puede representarse gráficamente por

una colección de nodos o vértices, asociando un nodo a cada elemento de $\mathbf{X}$. Estos nodos pueden conectarse por arcos, indicando las relaciones existentes entre los mismos. Un arco entre los nodos X_i y X_j se denotará mediante L_{ij}. Así mismo, el conjunto de todos los arcos se denotará por $\mathrm{L} = \left\{ \mathrm{L}_{ij} / X_i \text{ y } X_j \text{ están conectados} \right\}$. Por tanto, un grafo puede definirse mediante el conjunto de nodos, $\mathbf{X}$, y las relaciones entre los mismos, L. Los términos grafo y red se emplearán como sinónimos en este trabajo.

Definición. Un grafo es un par de conjuntos $\mathrm{G} = (\mathbf{X}, \mathrm{L})$ donde $\mathbf{X} = \{X_1, \cdots, X_n\}$ es un conjunto finito de elementos (nodos) y L es un conjunto de arcos, es decir, un subconjunto de pares ordenados de elementos distintos de $\mathbf{X}$.

El concepto de grafo puede definirse de forma más general. Por ejemplo, puede permitirse que dos nodos estén conectados por más de un arco o, incluso, que un nodo esté conectado consigo mismo. Sin embargo, al utilizar los grafos para representar un conjunto de variables (nodos), y unas relaciones de dependencia entre ellas (arcos) no será necesario que dos nodos estén unidos por más de un arco, o que un arco una a un nodo consigo mismo. Los arcos de un grafo pueden ser dirigidos o no dirigidos, dependiendo de si se considera o no el orden de los nodos.

- **Arco dirigido.** Dado un grafo $\mathrm{G}=(\mathbf{X},\mathrm{L})$, si $\mathrm{L}_{ij}\in\mathrm{L}$ y $\mathrm{L}_{ji}\notin\mathrm{L}$, el arco L_{ij} entre los nodos X_i y X_j se denomina dirigido y se denota mediante $X_i\rightarrow X_j$.
- **Arco no dirigido.** Dado un grafo $\mathrm{G}=(\mathbf{X},\mathrm{L})$, si $\mathrm{L}_{ij}\in\mathrm{L}$ y $\mathrm{L}_{ji}\in\mathrm{L}$, el arco L_{ij} se denomina no dirigido y se denota mediante X_i-X_j o X_j-X_i.
- **Grafo dirigido y no dirigido.** Un grafo en el cual todos los arcos son dirigidos se denomina grafo dirigido, y un grafo en el que todos sus arcos son no dirigidos se denomina no dirigido. Por tanto, en un grafo dirigido es importante el orden del par de nodos que definen cada arco, mientras que en un grafo no dirigido, el orden carece de importancia.
- **Camino entre dos nodos.** Un camino del nodo X_i al nodo X_j es un sucesión de nodos $\left(X_{i_1},\cdots,X_{i_r}\right)$, comenzando en $X_{i_1}=X_i$ y finalizando en $X_{i_r}=X_j$, de forma que existe un arco del nodo X_{i_k} al nodo $X_{i_{k+1}}$, $k=1,\cdots,r-1$. La longitud del camino, $(r-1)$, se define como el número de arcos que contiene.
- **Camino cerrado.** Un camino $\left(X_{i_1},\cdots,X_{i_r}\right)$ se dice que es cerrado si el nodo inicial coincide con el final, es decir, $X_{i_1}=X_{i_r}$.
- **Padre e hijo**. Cuando existe un arco dirigido, $X_i\rightarrow X_j$, del nodo X_i al nodo X_j, entonces se dice que el nodo X_i es un padre del nodo X_j, y que el nodo X_j es un hijo de X_i. El conjunto de los padres de un nodo X_i se denota por Pa_i.

- **Ciclo.** Un ciclo es un camino cerrado en un grafo dirigido.
- **Grafos cíclicos y acíclicos.** Un grafo dirigido se denomina cíclico si contiene al menos un ciclo; en caso contrario se denomina grafo dirigido acíclico.

2.8 Modelos definidos mediante grafos

La estructura de dependencias e independencias y, por tanto, la factorización asociada al modelo probabilístico, puede describirse mediante un grafo. Estos modelos de dependencia se conocen como modelos definidos gráficamente, y tienen como ejemplos más importantes a las redes Bayesianas.

Una ventaja de utilizar modelos gráficos para construir un modelo probabilístico es que estos modelos definen una factorización de la función de probabilidad como producto de funciones de probabilidad condicionada que determinan la estructura cualitativa del modelo probabilístico. Generalmente, estas funciones condicionadas contienen un número menor de variables que la función de probabilidad conjunta y, por tanto, el proceso de definición del modelo probabilístico es más sencillo.

Una vez que se conoce la estructura cualitativa del modelo probabilístico (la factorización de la función de probabilidad), la estructura cuantitativa de un modelo particular se define mediante la asignación de valores numéricos a los parámetros asociados a las funciones de probabilidad condicionada que intervienen en la factorización del modelo. Estos

valores han de ser definidos por algún experto, o estimados a partir de un conjunto de datos.

Por tanto, si la estructura cualitativa del modelo es desconocida, que es el caso habitual en la práctica, entonces tanto la estructura cualitativa, como la cuantitativa (los parámetros) han de ser estimadas a partir del conjunto de datos disponible.

2.9 Discretización

El proceso de discretización consiste en transformar los valores de una variable continua en un conjunto de valores discretos. Un algoritmo de discretización divide el dominio de la variable contínua en un número finito de intervalos disjuntos que cubran completamente el dominio. Luego se asigna a cada observación de la variable contínua el valor correspondiente al intervalo que lo contiene. Los métodos de discretización se dividen en

- **Métodos Locales o Globales.** Los métodos globales utilizan todas las observaciones para el proceso de discretización; a diferencia de los métodos locales que sólo utilizan un subconjunto de las mismas.
- **Métodos Supervisados o No Supervisados.** Los métodos de discretización pueden utilizar la información de la clase a la que pertenece cada observación (métodos supervisados) para mejorar la calidad de la discretización o no tomar en

consideración esta información (métodos no supervisados).

- **Métodos Estáticos o Dinámicos.** Los métodos estáticos son aquellos que discretizan cada variable independientemente de las demás. Los métodos dinámicos son aquellos que discretizan todas las variables simultáneamente tratando de utilizar las dependencias existentes entre ellas.
- **Métodos "top-down" (separación) o "bottom-up" (agrupamiento).** Los métodos "top-down" comienzan con una lista vacía de puntos de corte. Estos se van agregando dividiendo los intervalos conforme el proceso de discretización se realiza. Los métodos "bottom-up" comienzan con una lista completa de todos los valores continuos de la variable como puntos de corte. Estos se van eliminando juntando los intervalos conforme el proceso de discretización se realiza.

Se menciona, muy brevemente, algunos de los métodos de discretización encontrados frecuentemente en la literatura.

2.9.1 Método igual longitud

Es el más simple de los métodos de discretización no supervisados. Se calcula el valor máximo y mínimo de la variable que se desea discretizar y se divide el rango en k intervalos de igual longitud.

2.9.2 Método igual frecuencia

Se trata de otro método de discretización no supervisado. Tomando en consideración el número de valores de la variable que se desea discretizar, ésta se divide en intervalos que contengan el mismo número de observaciones. Experimentalmente se demostró que tomando $k = \max\{1, \log l\}$, donde l es el número de valores diferentes, se obtienen buenos resultados.

2.9.3 Discretización por mínima entropía

El proceso de discretización por mínima entropía es un método "top-down" supervisado propuesto por Fayyad e Irani [17]. Este método selecciona recursivamente los puntos de corte mediante un algoritmo de minimización de la entropía usando el criterio de Longitud de Descripción Mínima (LDM) [41] como criterio de parada.

2.9.4 ChiMerge

Este método de discretización "bottom-up" supervisado fue diseñado por Kerber [27] quien define un criterio para medir la calidad del proceso de discretización en el que las frecuencias relativas de clase deben ser consistentes entre intervalos. ChiMerge usa el estadístico de prueba χ^2 para determinar la independencia de la clase en dos intervalos adyacentes, juntándolos si éstos son independientes y separándolos en caso contrario.

El algoritmo ChiMerge tiene dos etapas. En la primera etapa se ordenan las observaciones de la variable que desea discretizar y se inicia el proceso colocando cada observación en su propio intervalo. La segunda etapa es un proceso de agrupamiento, que contiene dos pasos que se repiten hasta que no sean necesarios más agrupamientos. En el primer paso se calcula el valor del estadístico χ^2 para cada par de intervalos adyacentes. En el segundo se agrupan aquellos intervalos con menor valor para el χ^2. Se repite el proceso hasta que todos los valores χ^2 sean mayores que cierto umbral asociado al nivel de significación de la prueba. Este es el método de discretización usado en la presente tesis.

3 REDES BAYESIANAS

3.1 Introducción

Las redes Bayesianas proporcionan una representación gráfica para un conjunto de variables aleatorias y para las relaciones existentes entre ellas. La estructura de la red permite especificar la función de probabilidad conjunta de estas variables como el producto de funciones de probabilidad condicionadas, por lo general, más sencillas. En esta sección se definen formalmente las redes Bayesianas y se mencionan algunos métodos relacionados con su estimación.

3.2 Red Bayesiana

Una red Bayesiana para un conjunto de variables aleatorias $\mathbf{X} = \{X_1, \cdots, X_n\}$ es un par $\mathrm{B} = (\mathrm{G}, \mathrm{P}(\Theta))$, donde G es un gráfico acíclico dirigido, cuyos nodos se encuentran en correspondencia uno a uno con las variables en $\mathbf{X}$, y P es un conjunto de funciones de probabilidad local definidas por un conjunto de parámetros Θ. Se usa Pa_i y pa_i para denotar, respectivamente, a los padres y las configuraciones de los padres del nodo X_i en G. La función de probabilidad conjunta representada por la estructura G está dada por

$$p(\mathbf{X}) = p(X_1, \cdots, X_n) = \prod_{i=1}^{n} p(X_i / \mathrm{Pa}_i) \tag{3.1}$$

Las funciones de probabilidad local P son las funciones de probabilidad correspondientes a los términos de la ecuación (3.1). Un ejemplo de red Bayesiana se presenta en la figura 3.1. La función de probabilidad conjunta representada en esta red es $p(X_1, \cdots, X_5) = p(X_1 / X_2, X_5)\, p(X_2)\, p(X_3 / X_5)\, p(X_4 / X_3, X_5)\, p(X_5)$.

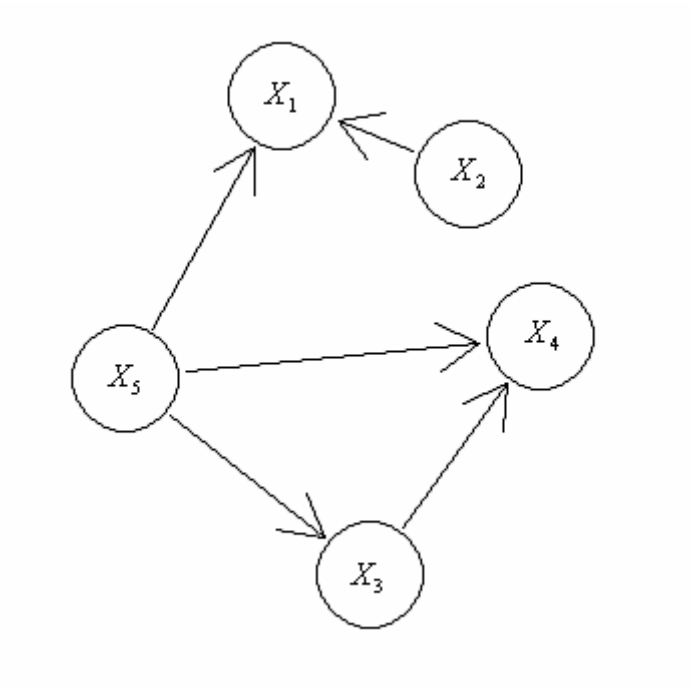

Figura 3.1: Red Bayesiana

Una red Bayesiana puede usarse para calcular una probabilidad de interés usando métodos para el proceso de inferencia exacta y aproximada [13, 25, 30]. El proceso de

inferencia exacta en redes Bayesianas es NP-hard (*non-polinomial-hard*) [11, 22], este problema se debe a la presencia de ciclos no dirigidos en la red. En general los algoritmos se consideran manejables cuando el tiempo necesario para ejecutar el algoritmo es polinomial en el número de parámetros, es decir, el número de nodos en la red [16]. Por ejemplo, un problema cuyo algoritmo más eficiente tiene tiempo de ejecución exponencial en el número de parámetros es NP-hard.

3.3 Estimación de los parámetros de una red Bayesiana

En esta sección se supone que la estructura de la red Bayesiana es conocida y que el conjunto de datos esta completo. Sea G^h que denota la hipótesis que la función de probabilidad de $\mathbf{X} = \{X_1, \cdots, X_n\}$ puede factorizarse de acuerdo a la estructura G. Sea $\theta_{\mathrm{G}} = \{\theta_1, \cdots, \theta_n\}$ un conjunto donde θ_i es el vector de parámetros de la función de probabilidad local $p\left(x_i / \mathrm{pa}_i, \theta_i, \mathrm{G}^h\right)$. Ahora es posible escribir la función de probabilidad conjunta como

$$p\left(\mathbf{X} / \theta_{\mathrm{G}}, \mathrm{G}^h\right) = \prod_{i=1}^{n} p\left(x_i / \mathrm{pa}_i, \theta_i, \mathrm{G}^h\right) \tag{3.2}$$

El problema de estimación de parámetros en una red Bayesiana se reduce a calcular la función de probabilidad posterior $p\left(\theta_{\mathrm{G}} / \mathrm{D}, \mathrm{G}^h\right)$, donde D representa el conjunto de datos de entrenamiento. Suponiendo que los parámetros θ_i son mutuamente

independientes, entonces

$$p\left(\theta_{\mathrm{G}} / \mathrm{D}, \mathrm{G}^{h}\right)=\prod_{i=1}^{n} p\left(\theta_{i} / \mathrm{D}, \mathrm{G}^{h}\right) \tag{3.3}$$

En otras palabras, cada vector de parámetros puede estimarse independientemente de los otros.

La estimación de parámetros para conjuntos de datos completos es discutida por Spiegelhalter y Lauritzen [39] y Buntine [5]. La estimación de los parámetros θ_i es más conveniente para familias exponenciales y distribuciones de probabilidad a priori conjugadas. Heckerman [22] discute la estimación Bayesiana de parámetros, usada por Friedman [18] con el nombre de suavización de parámetros.

3.3.1 Suavización de parámetros

Se supone que todas las variables X_i en la red Bayesiana G tienen distribución multinomial (si el conjunto de datos contiene variables continuas éstas se discretizan). Cada función de probabilidad local P_i asociada con la variable X_i es una colección de distribuciones multinomiales P_{ij}, una distribución para cada configuración de los padres Pa_i. Luego

$$p\left(x_{i}^{k} / \mathrm{pa}_{i}^{j}, \theta_{i}, \mathrm{G}^{h}\right)=\theta_{ijk}>0 \tag{3.4}$$

denota la probabilidad que la variable X_i se encuentre en su configuración k y sus padres en la configuración j. El número de configuraciones de la variable X_i es r_i y el número de configuraciones de sus padres es q_i. Los parámetros asociados con la variable X_i son $\theta_i = \left(\theta_{ij}\right)_{j=1}^{q_i}$. El subíndice k empieza en 2 ya que los parámetros θ_{ij1} pueden calcularse usando $\theta_{ij1} = 1 - \sum_{k=2}^{r_i} \theta_{ijk}$. El vector de parámetros asociados con cada función de probabilidad local P_{ij} se denota por $\theta_{ij} = \left(\theta_{ij2}, \cdots, \theta_{ijr_i}\right)$.

Como ya se mencionó el problema de estimación de los parámetros de una red Bayesiana, dada una estructura de red y un conjunto de datos de entrenamiento D, es el de encontrar la distribución posterior $p\left(\theta/\mathrm{D}, \mathrm{G}^h\right)$. Esta distribución puede calcularse eficientemente considerando que el conjunto de datos esta completo e independencia de parámetros. Es decir

$$p\left(\theta/\mathrm{D}, \mathrm{G}^h\right) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} p\left(\theta_{ij}/\mathrm{D}, \mathrm{G}^h\right)$$

Esto permite actualizar cada vector de parámetros independientemente. Además se supone que cada vector θ_{ij} tiene distribución a priori Dirichlet

$$p\left(\theta_{ij}\right)=\mathrm{Dir}\left(\theta_{ij}/\alpha_{ij1},\cdots,\alpha_{ijr_i}\right)=\frac{\Gamma\left(\alpha_{ij}\right)}{\prod_{k=1}^{r_i}\Gamma\left(\alpha_{ijk}\right)}\prod_{k=1}^{r_i}\theta_{ijk}^{\alpha_{ijk}-1} \tag{3.5}$$

donde $\alpha_{ij}=\sum_{k=1}^{r_i}\alpha_{ijk}$ y $\alpha_{ijk}>0$. Puesto que la distribución Dirichlet es la a priori conjugada de la distribución Multinomial se tiene

$$p\left(\theta_{ij}/\mathrm{D},\mathrm{G}^h\right)=\mathrm{Dir}\left(\theta_{ij}/\alpha_{ij1}+\mathrm{N}_{ij1},\cdots,\alpha_{ijr_i}+\mathrm{N}_{ijr_i}\right) \tag{3.6}$$

donde N_{ijk} es el número de casos en D en el que la variable X_i esta en la configuración k, $X_i=x_i^k$, y los padres de X_i están en la configuración j, $\mathrm{Pa}_i=\mathrm{pa}_i^j$.

Dado un conjunto de datos completo D y una estructura de red G^h pueden estimarse los parámetros θ usando el valor esperado de la distribución posterior. Usando la ecuación (3.4) se tiene que la probabilidad de un caso no observado $\mathbf{x}_{\mathrm{N}+1}$ es

$$p\left(\mathbf{x}_{\mathrm{N}+1}/\mathrm{D},\mathrm{G}^h\right)=\mathrm{E}_{p\left(\theta/\mathrm{D},\mathrm{G}^h\right)}\left[\prod_{i=1}^{n}\theta_{ijk}\right]$$

donde E denota el valor esperado. Usando la ecuación (3.6) y las propiedades de la distribución Dirichlet pueden estimarse los parámetros θ_{ijk} como sigue

$$p\left(\mathbf{x}_{\mathrm{N}+1}/\mathrm{D},\mathrm{G}^h\right)=\prod_{i=1}^{n}\frac{\alpha_{ijk}+\mathrm{N}_{ijk}}{\alpha_{ij}+\mathrm{N}_{ij}} \tag{3.7}$$

donde $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$.

3.4 Estimación de la estructura de la red Bayesiana

Existen dos metodologías para el problema de estimación de la estructura de una red Bayesiana. La primera, llamada *búsqueda y score*, utiliza un criterio para medir que tanto la estructura de la red estima el conocimiento a priori y un método de búsqueda para encontrar el modelo que mejor estime el conjunto de datos. La segunda metodología, llamada *análisis de dependencia*, mide el grado de relación de estas dependencias usando alguna prueba estadística o la cantidad de información mutua. Generalmente, la primera metodología tiene menor tiempo de complejidad pero puede no encontrar la mejor solución debido a su naturaleza heurística. La segunda metodología de algoritmos es generalmente asintóticamente correcta cuando la función de probabilidad cumple ciertas condiciones.

3.4.1 Selección del modelo por búsqueda y score

Esta sección presenta algunos ejemplos de las medidas de calidad usadas para calcular el score de una estructura de red, y los algoritmos de búsqueda sobre el espacio posible de estructuras de red.

A) Medidas de calidad

- **Medidas de calidad Bayesianas.** Estas medidas están relacionadas con la estadística Bayesiana. La idea básica es asignar a cada red un valor en función de su probabilidad posterior $p(\mathrm{B}/\mathrm{D})$. Un criterio frecuente es usar el logaritmo de la función de probabilidad posterior

$$\log p(\mathrm{B}/\mathrm{D}) = \log p(\mathrm{B}) + \log p(\mathrm{D}/\mathrm{B}) \tag{3.8}$$

El logaritmo es usado por conveniencia numérica. Este criterio tiene dos componentes: el logaritmo de la a priori y el log de la verosimilitud marginal. Un criterio equivalente usado es

$$\log\left(\frac{p(\mathrm{B}/\mathrm{D})}{p(\mathrm{B}_0/\mathrm{D})}\right) = \log\left(\frac{p(\mathrm{B}/\mathrm{D})}{p(\mathrm{B}_0/\mathrm{D})}\right) + \log\left(\frac{p(\mathrm{D}/\mathrm{B})}{p(\mathrm{D}/\mathrm{B}_0)}\right) \tag{3.9}$$

El ratio $\frac{p(\mathrm{D}/\mathrm{B})}{p(\mathrm{D}/\mathrm{B}_0)}$ es conocido como el *factor de Bayes*, donde B_0 es alguna hipótesis seleccionada como referencia.

- **Medidas de mínima longitud de codificación.** El score esta basado en el principio LDM [34] y la aplicación a las redes Bayesianas fue desarrollado por Bouckaert [3], Lam y Bacchus [29] y Suzuki [40]. Estos autores argumentan que usando solo la función de probabilidad posterior resulta en un criterio que refiere redes con gráficos completos, por lo que se debe agregar un factor que penalize el tamaño de la red.

$$q(\mathrm{B},\mathrm{D}) = \frac{\log \mathrm{N}}{2} \| \mathrm{B} \| - \mathrm{LL}(\mathrm{B}/\mathrm{D})$$

Donde $\| \mathrm{B} \|$ representa el tamaño de la red y LL el logaritmo de la función de verosimilitud.

- **Medidas de información teórica.** Otra forma de medir la calidad de una red es usando las medidas de información. Estas medidas pueden considerarse como una generalización de la medida LDM. Las más conocidas son

  Criterio de información por máxima verosimilitud. La medida es el logaritmo de la verosimilitud de una red Bayesiana, dado un conjunto de datos de entrenamiento. A diferencia de otras medidas de información teórica, esta no contiene penalización debido al tamaño de la red.

  $$q(\mathrm{B},\mathrm{D}) = \mathrm{LL}(\mathrm{B}/\mathrm{D})$$

  Criterio de información de Akaike. [1] Esta medida no es consistente ya que el mejor modelo no se encuentra entre las que reciben lo mayores scores [36].

  $$q(\mathrm{B},\mathrm{D}) = \mathrm{LL}(\mathrm{B}/\mathrm{D}) - \| \mathrm{B} \|$$

  Criterio de información Bayesiana. También conocido como criterio de información de Schwarz [36, 22] es fácil de usar y no requiere la evaluación de la función de probabilidad a priori. Consecuentemente, se trata de un criterio práctico usado en circunstancias apropiadas. Cuando se aplica a redes con distribución multinomial para las variables este criterio

es parecido al criterio LDM diferenciándose por un signo negativo.

$$q(\mathrm{B},\mathrm{D}) = \mathrm{LL}(\mathrm{B}/\mathrm{D}) - \frac{\log \mathrm{N}}{2} \| \mathrm{B} \|$$

B) Algoritmos de búsqueda del modelo

- **Algoritmo K2.** Cooper y Herskovits [12] describen el uso de un algoritmo de búsqueda *greedy* para identificar la estructura más probable dado un conjunto de datos de prueba. El algoritmo supone que existe un ordenamiento entre los nodos. El algoritmo empieza con una red vacía. Para cada X_i se consideran todos los nodos que podrían añadirse al conjunto de actual de padres. El nodo candidato que maximice la función de score local q_i es seleccionado. Si además para este nodo el conjunto actual de padres de X_i incrementa la función de score se agrega a pa_i y se continua la búsqueda para la siguiente variable. Si el nodo no incrementa la función de score se supone que se han encontrados todos los padres de X_i y el algoritmo empieza la búsqueda de los padres de X_{i+1}.
- **Algoritmo de Buntine.** Un algoritmo que no requiere un ordenamiento de los nodos fue propuesto por Buntine [5]. Este algoritmo comienza con un conjunto vacío de padres y en cada paso se agrega un nuevo enlace de tal manera que no se formen ciclos y que se maximice el incremento de la calidad. El proceso se repite hasta que no sea posible incrementar más la calidad o hasta que se haya

encontrado una red completa.

- **Algoritmo CB.** Singh y Voltara, [38] propusieron una extensión al algoritmo K2 llamado CB. Este algoritmo utiliza pruebas de independencia condicional para generar una "buena" ordenación de nodos a partir del conjunto de datos y luego usa el algoritmo K2 para generar la red Bayesiana a partir de una muestra de entrenamiento D usando el ordenamiento anterior. Empieza con un gráfico dirigido completo con todas las variables, luego remueve aquellos arcos entre nodos adyacentes que son condicionalmente independientes. CB orienta los ejes en el gráfico y obtiene un nuevo ordenamiento de las variables. Luego se usa el algoritmo K2 para construir la red correspondiente. El algoritmo repite este proceso hasta que el comportamiento predictivo de la red resultante deje de incrementarse.

3.4.2 Selección del modelo usando análisis de dependencias

La estimación de la estructura de una red Bayesiana usando el análisis de dependencia se basa en el uso de pruebas sobre subconjuntos de arcos en la red. Las metodologías estadísticas están basadas en las pruebas χ^2 [40]. Las metodologías basadas en medidas de información son investigadas por Cheng et al. [7, 8].

Método de Chow y Lui para estimar estructuras de árbol

El cálculo de las pruebas de independencia condicional puede ser mucho más eficiente si se imponen algunas restricciones sobre el gráfico. Chow y Lui [10] proponen un algoritmo para estimar las relaciones de dependencia entre un conjunto de variables discretas usando una estructura de árbol. Un grafico acíclico dirigido es un árbol si cada variable X_i tiene exactamente un padre, a excepción de una variable que no tiene padres (esta variable es conocida como raíz). Un ejemplo de red Bayesiana que tiene estructura de árbol se muestra en la figura 2.2.

En la representación gráfica de las relaciones de dependencia Chow y Lui [10] asignan pesos a los arcos usando el criterio de información mutua. La información mutua entre las variables discretas X_i y X_j se define como

$$\mathrm{I}\left(X_i ; X_j\right) = \sum_{x_i} \sum_{x_j} p\left(x_i, x_j\right) \log \frac{p\left(x_i, x_j\right)}{p\left(x_i\right) p\left(x_j\right)} \tag{3.10}$$

Este procedimiento reduce el problema de construir un árbol de máxima verosimilitud para encontrar un *árbol expandido de máximo peso* en un gráfico. El problema esta en seleccionar un conjunto de arcos que constituyan un árbol y que la suma de los pesos asociados a estos arcos sea maximizada. Chow y Lui [10] demostraron que la función de probabilidad representada por un árbol, construida usando su algoritmo es una

aproximación óptima de la verdadera distribución de probabilidad.

Algoritmo 1. Chow y Lui

1. Calcular la información mutua $\mathrm{I}(X_i; X_j)$ con $i < j$ donde i , $j = 1, 2, \cdots, n$. Asignar este valor como peso al arco que conecta las variables X_i y X_j.
2. Ordenar $\mathrm{I}(X_i; X_j)$ de mayor a menor.
3. Considerar un árbol inicial en el que no existen arcos entre las variables.
4. Asignar los dos arcos de mayor peso al árbol anterior.
5. Examinar el siguiente arco de mayor peso, y añadirla al árbol a no ser que forme un ciclo, en cuyo caso se descarta y se examina el siguiente arco de mayor peso.
6. Repetir 5 hasta seleccionar $n-1$ arcos.
7. Transformar el árbol no dirigido resultante en uno dirigido, escogiendo una variable como raíz, para a continuación direccionar el resto de arcos.

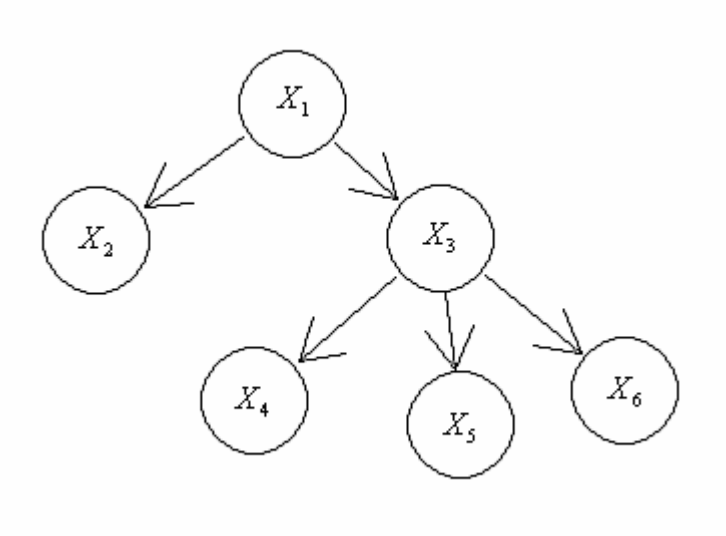

Figura 3.2: Red Bayesiana con estructura de árbol

4 CLASIFICADORES POR REDES BAYESIANAS

4.1 Introducción

Suponga que $X_1,\cdots,X_n$ son variables aleatorias que permiten predecir el valor de una variable C llamada clase. De acuerdo al teorema de Bayes, la probabilidad de que una observación pertenezca a la clase c es

$$p\left(C=c/x_1,\cdots,x_n\right)=\frac{p\left(C=c\right)p\left(x_1,\cdots,x_n/C=c\right)}{p\left(x_1,\cdots,x_n\right)} \tag{4.1}$$

El proceso de clasificación se realiza asignando la observación en aquella clase que tenga la mayor probabilidad usando la ecuación anterior. Las redes Bayesianas pueden ayudarnos a simplificar la representación de la función de probabilidad conjunta $p\left(X_1,\cdots,X_n/C\right)$ considerando las relaciones de dependencia que existen entre las variables.

4.2 Redes Bayesianas como clasificadores

Como ya se mencionó, una red Bayesiana para $\mathbf{X}=\left\{X_1,\cdots,X_n\right\}$ es un par $\mathrm{B}=\left(\mathrm{G},\mathrm{P}\left(\Theta\right)\right)$ donde G es un gráfico acíclico dirigido y P representa un conjunto de funciones de probabilidad local que tienen asociadas un conjunto de parámetros Θ. De

esta forma, una red Bayesiana B define una función de probabilidad única sobre $X_1, \cdots, X_n$ dada por

$$p_{\mathrm{B}}(\mathbf{X}) = p_{\mathrm{B}}(X_1, \cdots, X_n) = \prod_{i=1}^{n} p_{\mathrm{B}}(X_i / \mathrm{Pa_i}) = \prod_{i=1}^{n} \theta_{X_i / \mathrm{Pa_i}} \tag{4.2}$$

El problema de estimación de redes Bayesianas puede describirse informalmente como, dada una muestra de entrenamiento $\mathrm{D} = \{\mathbf{u}_1, \cdots, \mathbf{u}_{\mathrm{N}}\}$ encontrar una red B que mejor la describa. La metodología común para este problema es introducir una función de score que evalúe cada red con respecto a D y luego seleccione la mejor de acuerdo a esta función. En general, este problema de optimización es no manejable [9].

El principio LDM realiza el proceso de estimación en términos de la compresión del conjunto de datos. Comúnmente hablando, el objetivo del investigador es encontrar un modelo que facilite la menor descripción del conjunto original de datos. La longitud de esta descripción tiene en cuenta la descripción del modelo por si mismo y la descripción del conjunto de datos según el modelo. En el contexto de estimación por redes Bayesianas, el modelo es la red. Dicha red B describe una función de probabilidad p_{B} sobre las observaciones que aparecen en el conjunto de datos. De acuerdo con el principio LDM, se debería escoger una red B tal que la longitud combinada de la descripción de la red y el conjunto de datos codificado, con respecto a p_{B}, sea minimizada.

Sea $\mathrm{B}=\left(\mathrm{G},\mathrm{P}\left(\Theta\right)\right)$ una red Bayesiana, y sea $\mathrm{D}=\left\{\mathbf{u}_1,\cdots,\mathbf{u}_\mathrm{N}\right\}$ una muestra de entrenamiento, donde cada $\mathbf{u}_i$ asigna un valor a todas las variables en $\mathbf{X}$. La función de score LDM para una red B dada una muestra de entrenamiento D, $\mathrm{MDL}\left(\mathrm{B}/\mathrm{D}\right)$, esta dada por

$$\mathrm{LDM}\left(\mathrm{B}/\mathrm{D}\right)=\frac{\log \mathrm{N}}{2}\left\|\mathrm{B}\right\|-\mathrm{LL}\left(\mathrm{B}/\mathrm{D}\right) \tag{4.3}$$

donde $\left\|\mathrm{B}\right\|$ es el número de parámetros en la red. El primer término representa la longitud necesaria para describir la red B. El segundo término es el negativo del logaritmo de la verosimilitud de B dado D

$$\mathrm{LL}\left(\mathrm{B}/\mathrm{D}\right)=\sum_{i=1}^{\mathrm{N}}\log p_\mathrm{B}\left(\mathbf{u}_i\right) \tag{4.4}$$

Cuanto mayor sea el logaritmo de la verosimilitud, más cerca estará B de modelar la función de probabilidad del conjunto de datos D. Sea $\hat{p}_\mathrm{D}\left(\cdot\right)$ la distribución empírica definida por la frecuencia de eventos en D, $\hat{p}_\mathrm{D}\left(\mathrm{A}\right)=\frac{1}{\mathrm{N}}\sum_j 1_\mathrm{A}\left(\mathbf{u}_j\right)$ para cada evento A, donde $1_\mathrm{A}\left(\mathbf{u}\right)=1$ si $\mathbf{u}\in\mathrm{A}$ y $1_A\left(\mathbf{u}\right)=0$ si $\mathbf{u}\notin\mathrm{A}$. Aplicando la ecuación (4.2) al logaritmo de la verosimilitud, ésta se descompone de acuerdo a la estructura de B

$$\mathrm{LL}(\mathrm{B}/\mathrm{D}) = \mathrm{N}\sum_{i=1}^{\mathrm{N}} \sum_{x_i, \mathrm{Pa}_i} \hat{p}_{\mathrm{D}}(x_i, \mathrm{Pa}_i) \log\left(\theta_{x_i/\mathrm{Pa}_i}\right) \tag{4.5}$$

Se puede demostrar que esta expresión es maximizada cuando

$$\theta_{x_i/\mathrm{Pa}_i} = \hat{p}_{\mathrm{D}}(x_i/\mathrm{Pa}_i) \tag{4.6}$$

Consecuentemente, si se tiene dos redes Bayesianas, $\mathrm{B} = (\mathrm{G}, \Theta)$ y $\mathrm{B}' = (\mathrm{G}, \Theta')$, que comparten la misma estructura G, y si Θ satisface la ecuación anterior, entonces $\mathrm{LL}(\mathrm{B}/\mathrm{D}) \geq \mathrm{LL}(\mathrm{B}'/\mathrm{D})$. Así, dada una estructura de red, existe una solución de forma cerrada para los parámetros tal que maximiza el score del logaritmo de la verosimilitud, llamada, ecuación (4.6). Además, desde que el primer término de la ecuación (4.3) no depende de los parámetros escogidos, esta solución minimiza el score LDM. Esta es una observación crucial desde que nos releva de la búsqueda entre las redes Bayesianas y entonces estimamos los parámetros utilizando las frecuencias observadas en el conjunto de datos. De aquí en adelante, a menos que se mencione lo contrario, se supone que los parámetros escogidos satisfacen la ecuación (4.6).

Los scores del logaritmo de la verosimilitud no son convenientes para estimar la estructura de una red, desde que tiende a favorecer estructuras de gráficos completas en la que cada variable esta conectada con las demás. Esto es no deseable, ya estas redes

requieren un número exponencial de parámetros, mucho de los cuales podrían tener una variancia bastante alta realizando predicciones bastante pobres. Así, los parámetros estimados en una red completa ajustaran bien un conjunto de datos de entrenamiento pero mostraran un pobre comportamiento sobre el conjunto de datos de prueba. Este problema llamado, sobreajuste, se evita mediante el score LDM. El primer término del score LDM, ecuación (4.3), regula la complejidad de las redes penalizando aquellas que contienen muchos parámetros. Así, el score LDM de una red muy grande podría ser bastante mayor al score de una red pequeña, aún cuando la primera red tengo un mejor comportamiento sobre el conjunto de datos. En la práctica, el score LDM regula el número de parámetros estimados y nos ayuda a prevenir el sobreajuste en el conjunto de datos de entrenamiento.

Usando el método descrito, uno puede estimar una red Bayesiana B, que codifique una función de probabilidad $p_{\mathrm{B}}\left(X_1,\cdots,X_n,C\right)$ a partir de una muestra de entrenamiento. Luego, el modelo resultante puede usarse para que, dada una observación $x_1,\cdots,x_n$, el clasificador basado en B devuelva la clase c que maximiza la probabilidad posterior $p_{\mathrm{B}}\left(c/x_1,\cdots,x_n\right)$. Este procedimiento esta justificado por la corrección asintótica del procedimiento de estimación Bayesiano. Dado un conjunto grande de datos, la red estimada puede aproximarse a la función de probabilidad. Aunque este argumento tiene base teórica, en la práctica es posible encontrar casos en los que el proceso de aprendizaje proporcione una red con un score LDM relativamente bueno pero que funcione bastante

mal como clasificador.

Para entender la posible discrepancia entre una buena exactitud predictiva y un buen score LDM, reexaminemos este último. Recordar que el término del logaritmo de la verosimilitud en la ecuación (4.3) es el que mide la calidad del modelo construido, y que $\mathrm{D} = \{ \mathbf{u}_1, \cdots, \mathbf{u}_\mathrm{N} \}$ denota el conjunto de datos de entrenamiento. En el contexto de la clasificación, cada $\mathbf{u}_i$ es una tupla de la forma $\left(x_1^i, \cdots, x_n^i, c^i \right)$ que asigna valores a las variables $X_1, \cdots, X_n, C$. Es posible rescribir el logaritmo de la verosimilitud como

$$\mathrm{LL}(\mathrm{B}/\mathrm{D}) = \sum_{i=1}^{\mathrm{N}} \log p_\mathrm{B}\left(c^i / x_1^i, \cdots, x_n^i \right) + \sum_{i=1}^{\mathrm{N}} \log p_\mathrm{B}\left(x_1^i, \cdots, x_n^i \right) \tag{4.7}$$

El primer término en esta ecuación mide cuan bien B estima la probabilidad de pertenencia a cada clase dadas las variables. El segundo término mide cuan bien B estima la función de probabilidad conjunta de las variables. Desde que la clasificación se determina por $p_\mathrm{B}\left(C / X_1, \cdots, X_n \right)$, solo el primer término esta relacionado con la precisión del clasificador. Desafortunadamente, este término es dominado por el segundo cuando existen muchas variables; mientras *n* crece, la probabilidad de cada asignación particular a $X_1, \cdots, X_n$ se vuelve pequeña, ya que el número de posibles asignaciones crece exponencialmente en n. Así, se espera que términos de la forma $p_\mathrm{B}\left(X_1, \cdots, X_n \right)$ produzcan valores cercanos a cero, y consecuentemente $-\log p_\mathrm{B}\left(X_1, \cdots, X_n \right)$ tome

valores grandes. Sin embargo, al mismo tiempo la probabilidad condicional de la clase permanece siendo más o menos la misma. Esto implica que un error relativamente grande en el término condicional de la ecuación (4.7) puede no reflejarse en el score LDM. De esta manera es posible no obtener un buen clasificador en el caso de existir muchas variables.

4.3 Clasificador Naive Bayes

El clasificador Naive Bayes y sus variantes se encuentran entre los algoritmos más conocidos para construir clasificadores de documentos de texto [28], filtración de correo electrónico [35], clasificación de galaxias [2] y reconocimiento de emociones [37]. A pesar de su simplicidad es comparable con clasificadores sofisticados como las redes neuronales y los árboles de decisión, ya que posee alta precisión y velocidad cuando es aplicada a conjuntos grandes de datos.

El clasificador Naive Bayes fue popularizado por Duda y Hart [15] gracias a su simplicidad, eficiencia y bajo error de clasificación. Este clasificador supone que todas las variables son condicionalmente independientes dado el valor de la clase. Este supuesto simplifica la representación de $p\left(X_1,\cdots X_n/C\right)$ así como su estimación a partir de la muestra de entrenamiento.

La estructura de este clasificador puede representarse usando una red Bayesiana (figura 4.1) en la que existe un nodo para la variable de clase C, que es padre de todas las variables y en la que no existen arcos entre las variables.

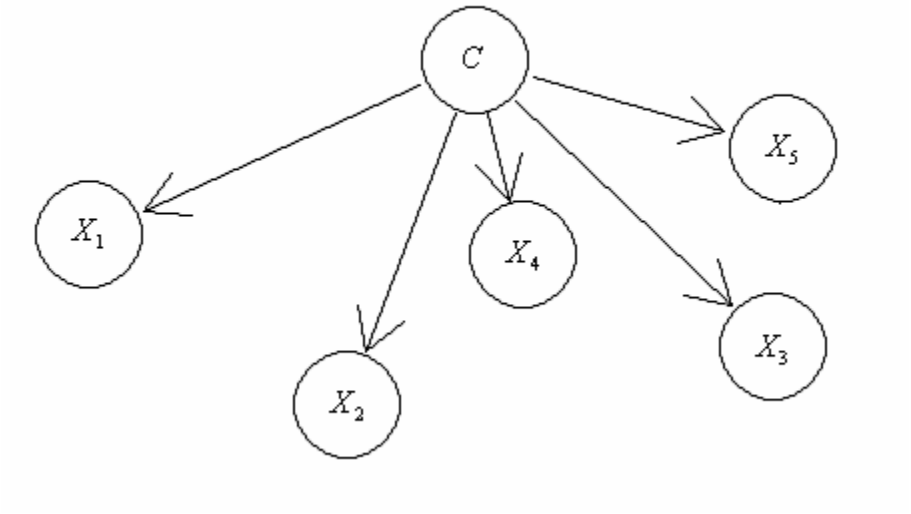

Figura 4.1: La estructura de una red Naive Bayes

El funcionamiento de este clasificador es algo sorprendente, ya que el supuesto de independencia no es realista. Considere un clasificador para la evaluación del riego en las solicitudes de crédito: es contra intuitivo ignorar las correlaciones entre edad, nivel de educación e ingreso. El ejemplo anterior lleva a la necesidad construir un clasificador que tome en consideración las relaciones de dependencia que existen en el conjunto de variables.

4.4 Extensiones del clasificador Naive Bayes

En esta sección se examinan dos metodologías que mantienen la estructura básica del clasificador Naive Bayes. Sin embargo para mejorar el comportamiento del clasificador, se propone aumentar la estructura de Naive Bayes con arcos entre las variables, cuando sean necesarios, desechando así el supuesto de independencia. Estas estructuras son llamadas redes aumentadas Naive Bayes y estos arcos, arcos aumentados [18].

En una estructura aumentada, un arco desde X_i hacia X_j implica que la influencia de X_i en la asignación de la variable de clase C también depende del valor de X_j. Adicionar el mejor conjunto de arcos aumentados es un problema no manejable, ya que equivale a encontrar la mejor red Bayesiana entre aquellas en las que C es una raíz. Así, aunque el comportamiento del clasificador Naive Bayes pudiera mejorarse, el esfuerzo computacional requerido puede no valer la pena. Sin embargo, imponiendo restricciones aceptables es posible encontrar el conjunto óptimo de arcos aumentados en un tiempo polinomial.

El propósito esta en encontrar una red Naive Bayes de árbol aumentado (TAN) en la que la variable de clase no tenga padres y las variables restantes tengan como padres a la variable de clase y a lo más alguna otra variable. Así, cada variable puede tener un arco aumentado dirigido hacia él. La red de la figura 4.2, es de hecho un modelo TAN. El

procedimiento para obtener estos arcos esta basado en el algoritmo de Chow y Liu [10] para estimar las relaciones de dependencia entre un conjunto de variables usando una estructura de árbol.

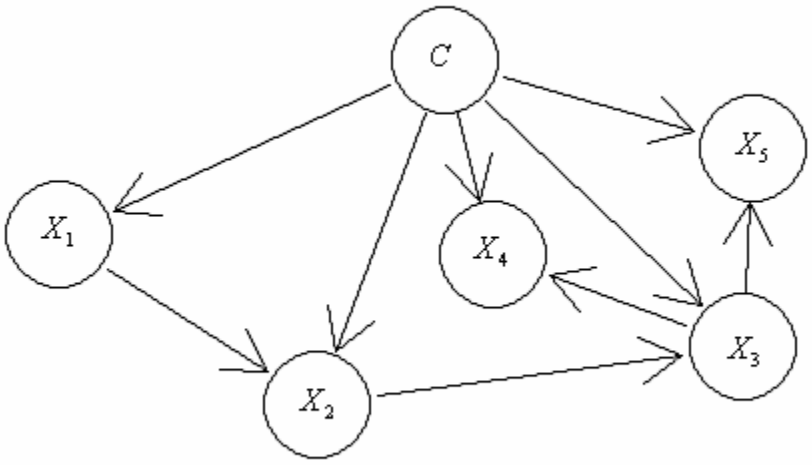

Figura 4.2: La estructura de una red TAN

Un gráfico acíclico dirigido sobre $X_1,\cdots,X_n$ es una árbol si Pa_i contiene exactamente un padre para todo X_i, excepto por una variable (esta variable es conocida como raíz). Una red con estructura de árbol puede describirse identificando los padres de cada variable. Una función $\pi:\{1,\cdots,n\}\mapsto\{0,\cdots,n\}$ se dice que define un árbol sobre $X_1,\cdots,X_n$ si existe exactamente un i tal que $\pi(i)=0$ (llamada la raíz del árbol) y no

existe alguna secuencia $i_1,\cdots,i_k$ tal que $\pi(i_j)=i_{j+1}$ para $i\le j<k$ y $\pi(i_k)=i_1$ (es decir no existen ciclos). Tal función define una red de árbol cuando $\mathrm{Pa}_i=\{X_{\pi(i)}\}$ si $\pi(i)>0$, y $\mathrm{Pa}_i=\varnothing$ si $\pi(i)=0$.

4.4.1 Clasificador Naive Bayes de árbol aumentado (TAN)

El algoritmo de Chow y Lui [10], a partir de una colección D de N observaciones de $X_1,\cdots,X_n$, construye un árbol $\mathrm{B_T}$ que maximiza $\mathrm{LL}(\mathrm{B_T}/\mathrm{D})$. Este resultado puede ahora adaptarse para encontrar la estructura TAN de máxima verosimilitud. Sea $X_1,\cdots,X_n$ un conjunto de variables y C la variable de clase. Diremos que B es un modelo TAN si $\mathrm{Pa}_C=\varnothing$ y existe una función π que define un árbol sobre $X_1,\cdots,X_n$ tal que $\mathrm{Pa}_{X_i}=\{C,X_{\pi(i)}\}$ si $\pi(i)>0$, y $\mathrm{Pa}_{X_i}=\{C\}$ si $\pi(i)=0$. El problema de optimización consiste en encontrar una función π que defina una estructura de árbol sobre $X_1,\cdots,X_n$ tal que el logaritmo de la verosimilitud sea maximizada.

Tal como se mostrara más adelante, el procedimiento que se denomina algoritmo TAN resuelve este problema de optimización. Este procedimiento sigue el enfoque general de Chow y Lui [10], excepto que en lugar de usar la información mutua entre dos variables, usa la información condicional mutua dada la variable de clase

$$\mathrm{I}\left(X_i ; X_j / C\right)=\sum_{x_i} \sum_{x_j} \sum_c p\left(x_i, x_j, c\right) \log \frac{p\left(x_i, x_j / c\right)}{p\left(x_i / c\right) p\left(x_j / c\right)} \tag{4.8}$$

Informalmente hablando, esta función mide la cantidad de información que X_j proporciona acerca de X_i cuando el valor de C es conocido. El algoritmo TAN se muestra a continuación.

Algoritmo 2. TAN

1. Calcular $\mathrm{I}\left(X_i ; X_j / C\right)$ con $i<j$ donde i, $j=1,2,\cdots,n$. Asignar este valor como peso al arco que conecta las variables X_i y X_j.
2. Ordenar $\mathrm{I}\left(X_i ; X_j / C\right)$ de mayor a menor.
3. Considerar un árbol inicial en el que no existen arcos entre las variables.
4. Asignar los dos arcos de mayor peso al árbol anterior.
5. Examinar el siguiente arco de mayor peso, y añadirla al árbol a no ser que forme un ciclo, en cuyo caso se descarta y se examina el siguiente arco con mayor peso.
6. Repetir 5 hasta seleccionar $n-1$ arcos.
7. Transformar el árbol no dirigido resultante en uno dirigido, escogiendo una variable como raíz, para a continuación direccionar el resto de arcos.
8. Construir un modelo TAN añadiendo un nodo etiquetado como C y posteriormente un arco desde C a cada variable predictora X_i.

Sea D una colección de N observaciones de $C, X_1, \cdots, X_n$. Se probara que el algoritmo TAN permite construir una red $\mathrm{B_T}$ que maximiza $\mathrm{LL}(\mathrm{B_T}/\mathrm{D})$. Reformulando la ecuación (4.5) en términos de la entropía

$$\mathrm{LL}(\mathrm{B_T}/\mathrm{D}) = -\mathrm{N}\sum_{X_i} \mathrm{H}_{\hat{p}_\mathrm{D}}(X_i/\mathrm{Pa}_i) = \mathrm{N}\sum_{X_i} \mathrm{I}_{\hat{p}_\mathrm{D}}(X_i;\mathrm{Pa}_i) - \mathrm{N}\sum_{X_i} \mathrm{H}_{\hat{p}_\mathrm{D}}(X_i)$$

Notar que $\mathrm{H}_{\hat{p}_\mathrm{D}}(X_i)$ es independiente de la elección de $\mathrm{B_T}$. Luego

$$\mathrm{LL}(\mathrm{B_T}/\mathrm{D}) = \mathrm{N}\sum_{X_i} \mathrm{I}_{\hat{p}_\mathrm{D}}(X_i;\mathrm{Pa}_i) + \text{término constante} \tag{4.9}$$

Luego, maximizar el logaritmo de la verosimilitud equivale a maximizar el término $\sum_{X_i} \mathrm{I}_{\hat{p}_\mathrm{D}}(X_i;\mathrm{Pa}_i)$. Ahora se especializa este término para los modelos TAN. Sea $\mathrm{B_T}$ un TAN definido por $\pi(\cdot)$. Desde que C no tiene padres, se tiene $\mathrm{I}_{\hat{p}_\mathrm{D}}(C;\mathrm{Pa}_C) = 0$. Como los padres de X_i están definidos por π, se tiene $\mathrm{I}_{\hat{p}_\mathrm{D}}(X_i;\mathrm{Pa}_i) = \mathrm{I}_{\hat{p}_\mathrm{D}}(X_i;X_{\pi(i)},C)$ si $\pi(i) > 0$ y $\mathrm{I}_{\hat{p}_\mathrm{D}}(X_i;\mathrm{Pa}_i) = \mathrm{I}_{\hat{p}_\mathrm{D}}(X_i;C)$ si $\pi(i) = 0$. De ahí, necesitamos maximizar el término

$$\sum_{i,\pi(i)>0} \mathrm{I}_{\hat{p}_\mathrm{D}}(X_i;X_{\pi(i)},C) + \sum_{i,\pi(i)=0} \mathrm{I}_{\hat{p}_\mathrm{D}}(X_i;C) \tag{4.10}$$

Es posible simplificar este término usando la identidad conocida como la regla de la

cadena para la información mutua, $\mathrm{I}_p\left(X;Y,Z\right)=\mathrm{I}_p\left(X;Z\right)+\mathrm{I}_p\left(X;Y/Z\right)$. Luego, puede rescribirse la ecuación (4.10) como

$$\sum_i \mathrm{I}_{\hat{p}_\mathrm{D}}\left(X_i;\,C\right)+\sum_{i,\pi(i)>0}\mathrm{I}_{\hat{p}_\mathrm{D}}\left(X_i;X_{\pi(i)}/C\right)$$

Notar que el primer término no es afectado por la selección de $\pi(i)$. Por lo tanto, es suficiente maximizar el segundo término. Note también que el modelo encontrado por el algoritmo TAN garantiza la maximización de este término, maximizando también la log verosimilitud, ya que la suma de los pesos en la estructura de árbol es máxima según el algoritmo de Chow y Lui [10].

4.4.2 Multiredes Bayesianas CL

El clasificador TAN supone que la relación entre las variables es la misma en cada una de las clases. Una inmediata generalización al método anterior considera diferentes estructuras de árbol para cada clase, y una colección de redes como clasificadores.

Para implementar esta idea se particiona el conjunto de datos de entrenamiento por clases. Luego, para cada clase c_i se construye una red Bayesiana B_i para las variables $X_1,\cdots,X_n$. La distribución de probabilidad resultante $p_{\mathrm{B}_i}\left(X_1,\cdots,X_n\right)$ aproxima la distribución conjunta de las variables, dada una clase especifica, esto es, $\hat{p}_\mathrm{D}\left(X_1,\cdots,X_n/C=c_i\right)$. La red Bayesiana para c_i es llamada una multired Bayesiana [19,

20]. Formalmente una multired es una tupla $\mathrm{M} = \left(p_C , \mathrm{B}_1 , \cdots , \mathrm{B}_k \right)$ donde p_C es una distribución sobre C y B_i es una red Bayesiana sobre $X_1 , \cdots , X_n$ para $1 \le i \le k$, donde k es el número de clases. Una multired M define una distribución conjunta

$$p_{\mathrm{M}} \left(C , X_1 , \cdots , X_n \right) = p_C \left(C \right) p_{\mathrm{B}_i} \left(X_1 , \cdots , X_n \right) \text{ cuando } C = c_i$$

Cuando se estima una multired, se toma $p_C \left(C \right)$ como la frecuencia de la variable de clase en el conjunto de datos de entrenamiento, esto es, $\hat{p}_C \left(C \right)$ y se construyen las redes B_i en la forma ya descrita. Una vez más, clasificamos escogiendo aquella clase que maximice la probabilidad posterior $p_{\mathrm{M}} \left(C / X_1 , \cdots , X_n \right)$. Particionando el conjunto de datos de acuerdo a la variable de clase, esta metodología asegura que las interacciones entre C y las variables sean tomadas en cuenta. La propuesta de una multired es estrictamente la generalización de una red TAN [18].

Puede usarse el algoritmo de Chow y Lui [10] separadamente para cada valor de la variable de clase. El resultado es una multired en la que cada red es un árbol.

Algoritmo 3. Multiredes CL

1. Dividir D en k particiones, $\mathrm{D}_1, \cdots, \mathrm{D}_k$, tal que D_i contiene todas las observaciones en D donde $C = c_i$.
2. Sea $p_C(c_i) = \widehat{p}_{\mathrm{D}}(c_i)$.
3. Aplicar el algoritmo 1 en D_i para construir B_i.

5 RESULTADOS Y DISCUSIONES

5.1 Conjuntos de datos

Se corrieron los experimentos sobre 20 conjuntos de datos listados en la tabla 1. Todos los conjuntos provienen del repositorio UCI Machine Learning con excepción de Mofn-3-7-10 y Corral. Estos dos conjuntos de datos experimentales fueron diseñadas por Jhon y Kohavi [26] para evaluar métodos de selección de subconjuntos de variables.

5.2 Preprocesamiento

En las metodologías discutidas en este trabajo se supone que todas las variables son discretas y que no existen valores perdidos en el conjunto de datos. Sin embargo, las variables continuas se encuentran presentes en muchos conjuntos de datos. Rodríguez [33] tiene una librería llamada dprep dentro del software estadístico R que incluye funciones para el preprocesamiento de conjuntos de datos. Usando esta librería se realizo el proceso de discretización de las variables continuas usando la función ChiMerge y eliminando las observaciones con valores perdidos usando la función na.omit.

Tabla 1. Descripción de los conjuntos de datos usados en el experimento

N	Datos	ChiMerge	Variables	Clases	Observaciones	
					Entrenamiento	Prueba
1	Australian	Sí	14	2	690	CV10
2	Breast	No	9	2	683	CV10
3	Chess	Sí	36	2	2130	1066
4	Cleve	Sí	13	2	296	CV10
5	Corral	No	6	2	128	CV10
6	Crx	Sí	15	2	653	CV10
7	Diabetes	Sí	8	2	768	CV10
8	Flare	No	10	2	1066	CV10
9	German	Sí	20	2	1000	CV10
10	Glass	Sí	9	6	214	CV10
11	Heart	Sí	13	2	270	CV10
12	Hepatitis	Sí	19	2	80	CV10
13	Iris	Sí	4	3	150	CV10
14	Letter	Sí	16	26	15000	5000
15	Monf-3-7-10	No	10	2	300	1024
16	Satimage	Sí	36	6	4435	2000
17	Segment	Sí	18	7	1540	770
18	Shuttle	Sí	9	7	43500	14500
19	Vehicle	Sí	18	4	846	CV10
20	Vote	No	16	2	435	CV10

5.3 Precisión del clasificador

La precisión de cada clasificador se estima usando el porcentaje de observaciones correctamente clasificadas. En el caso de los conjuntos de datos grandes, el proceso de estimación de la red y sus parámetros se realiza sobre la muestra de entrenamiento para luego evaluar la precisión del clasificador sobre la muestra de prueba. En el caso de los conjuntos de datos pequeños el proceso de estimación de la red y sus parámetros se

realiza mediante validación cruzada 10, es decir

- Se divide en conjunto de datos en 10 partes.
- Se aplica el proceso de estimación usando 9 de estas partes como muestra de entrenamiento y evaluando el clasificador con la parte restante (muestra de prueba).
- Se repite 10 veces el experimento, utilizando cada vez una parte distinta de las 10 como muestra de prueba y las nueve restantes para entrenamiento.

Las siguientes abreviaturas se usan para mencionar las diferentes metodologías de clasificación discutidas.

NB	Naive Bayes
TAN	Redes TAN
CL	Multiredes Bayesianas CL

5.4 Resultados

Los resultados obtenidos con los clasificadores NB, TAN y CL se encuentran en la tabla 2. Los valores en negrita identifican el clasificador con el que se obtuvo la mayor precisión en cada conjunto de datos. Los parámetros se estiman mediante tablas condicionales usando las frecuencias observadas en cada configuración. En general, con

cualquiera de los clasificadores anteriores, el porcentaje exitoso de predicción esta dentro de los valores reportados por otros clasificadores. Se construyen diagramas de dispersión que comparan los clasificadores mencionados. En este tipo de diagramas los puntos sobre la diagonal corresponden a conjuntos de datos en los que el clasificador que se encuentra en el eje vertical tuvo un mejor comportamiento predictivo en comparación al clasificador que se encuentra en el eje horizontal.

Tabla 2. Precisión de los clasificadores Naive Bayes, TAN y Multiredes CL usando frecuencias condicionales

N	Datos	NB	TAN	CL
1	Australian	**88.7**	85.3	83.7
2	Breast	**97.6**	96.5	96.9
3	Chess	87.1	**92.5**	91.7
4	Cleve	**87.2**	81.6	82.3
5	Corral	86.6	98.7	**98.8**
6	Crx	**89.1**	84.7	84.1
7	Diabetes	**83.5**	79.8	79.9
8	Flare	79.9	**82.8**	81.4
9	German	**81.7**	78.8	78.8
10	Glass	**79.1**	78.1	78.2
11	Heart	**87.9**	82.7	81.0
12	Hepatitis	92.5	**94.3**	94.0
13	Iris	**95.8**	95.7	95.1
14	Letter	74.2	87.9	**89.2**
15	Monf-3-7-10	86.4	91.0	**91.5**
16	Satimage	82.5	**88.7**	86.8
17	Segment	**92.1**	89.2	88.3
18	Shuttle	99.3	**99.9**	**99.9**
19	Vehicle	68.6	76.8	**77.0**
20	Vote	90.2	**93.6**	**93.6**

La figura 5.1 compara la precisión del clasificador NB versus TAN. Tal como puede observarse TAN domina a NB en la mitad de los conjuntos de datos. Considerando las relaciones de dependencia es posible construir mejores clasificadores. Los conjuntos de datos en las que TAN tuvo mejor comportamiento predictivo son aquellas que tienen muestra de prueba y de entrenamiento con la excepción de Segment.

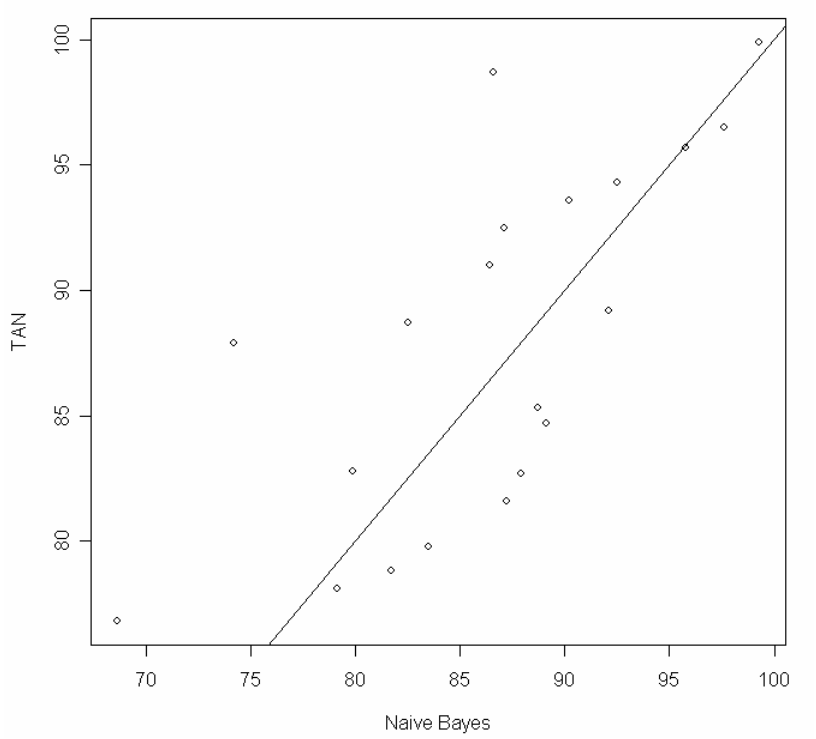

Figura 5.1: Diagramas de dispersión para la precisión obtenida con los clasificadores Naive Bayes y TAN

La figura 5.2 compara la precisión del clasificador NB versus CL. Al igual que TAN, CL es superior a NB en los mismos conjuntos de datos. Es muy probable que en estos conjuntos existan fuertes relaciones de dependencia entre las variables. La figura 5.3 compara la precisión del clasificador TAN versus CL. Como puede observarse estos clasificadores tienen un comportamiento predictivo bastante uniforme.

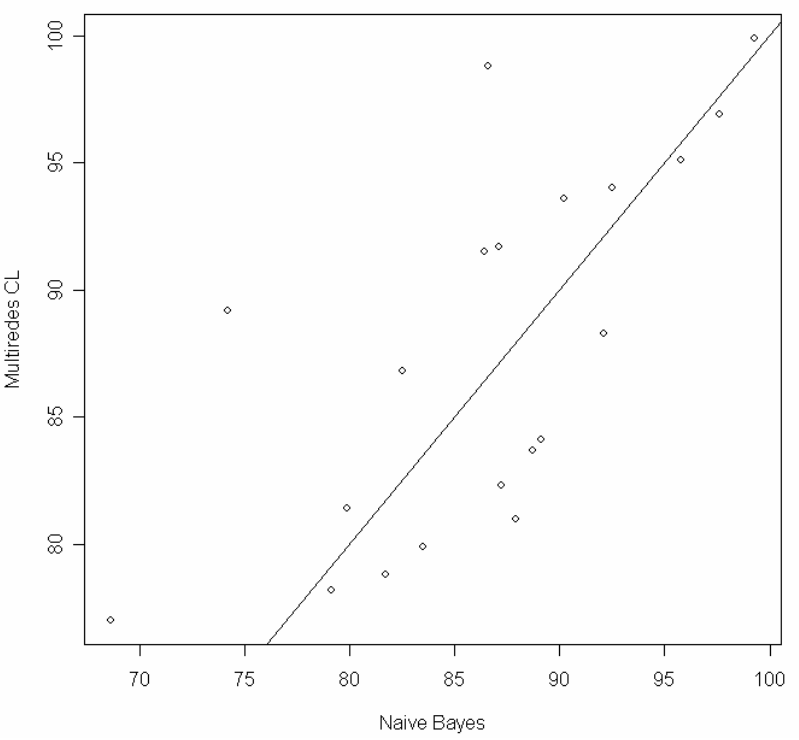

Figura 5.2: Diagramas de dispersión para la precisión obtenida con los clasificadores Naive Bayes y Multiredes CL

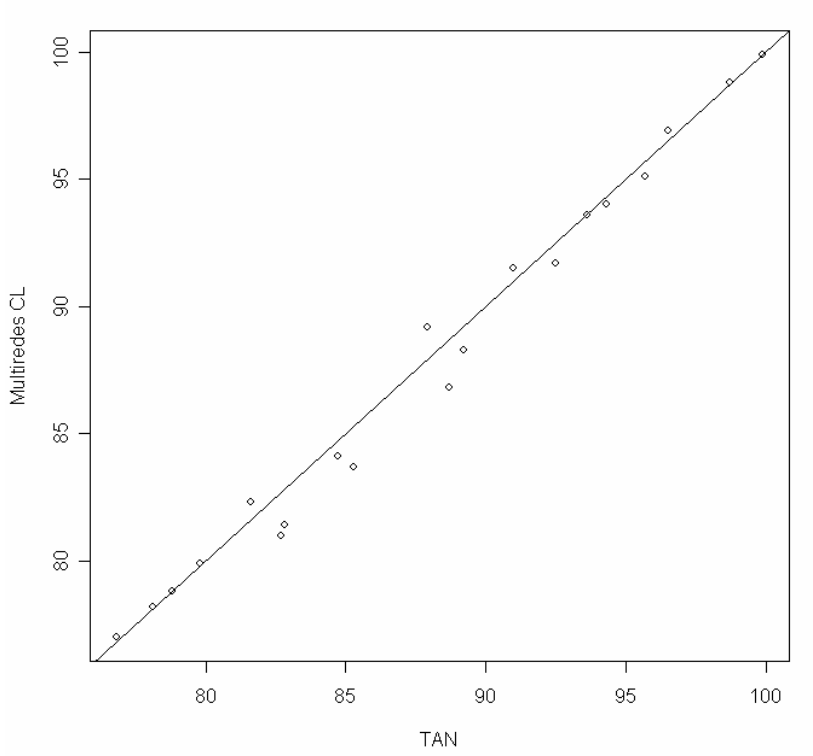

Figura 5.3: Diagramas de dispersión para la precisión obtenida con los clasificadores TAN y Multiredes CL

Para estimar los parámetros de una red Bayesiana se utilizan las frecuencias condicionales de la forma $\hat{p}_{\mathrm{D}}(X/\mathrm{Pa}_i)$. Para hacerlo se requiere particionar el conjunto de datos de entrenamiento de acuerdo a los posibles valores de Pa_i y calcular las frecuencias de X_i en cada partición.

Cuando algunas de estas particiones contienen muchas instancias, el estimado de la probabilidad condicional puede no ser confiable. Este problema no es grave en el caso del

clasificador Naive Bayes, debido a que éste particiona el conjunto de datos de acuerdo a la variable de clase y todos los valores que toma están adecuadamente representados en el conjunto de datos de entrenamiento. En las redes TAN, sin embargo, para cada variable evaluamos la probabilidad condicional dada la variable de clase y otra variable. Esto significa que el número de particiones es al menos tan grande como dos. Así, no es sorprendente encontrar estimados no fiables, especialmente en conjuntos de datos pequeños.

Para tratar este problema, se introduce una operación de suavización sobre los parámetros, tal como se mencionó en la sección 3.3, motivada por consideraciones Bayesianas. Se supone que todas las variables en la red Bayesiana tienen distribución multinomial y que cada vector de parámetros tiene distribución a priori Dirichlet.

Por lo tanto, para usar este método debemos escoger los parámetros a priori. Una elección razonable en caso de no disponer de información previa es la distribución uniforme. Después de pruebas iniciales, considerando $\alpha_{ij} = 1$, $\alpha_{ij} = 5$ y $\alpha_{ij} = 10$, se escogió el valor $\alpha_{ij} = 5$ en todos nuestros experimentos. Las siguientes abreviaturas se usan para mencionar las diferentes metodologías de clasificación usando suavización de parámetros.

SNB Naive Bayes con suavización de parámetros

STAN Redes TAN con suavización de parámetros

SCL Multiredes Bayesianas CL con suavización de parámetros

Tabla 3. Precisión de los clasificadores Naive Bayes, TAN y Multiredes CL usando frecuencias condicionales y la suavización de parámetros

N	Datos	NB	SNB	TAN	STAN	CL	SCL
1	Australian	88.7	**89.0**	85.3	**87.7**	83.7	**84.9**
2	Breast	**97.6**	**97.6**	96.5	**96.7**	**96.9**	95.8
3	Chess	87.1	**87.3**	92.5	**92.7**	**91.7**	91.6
4	Cleve	**87.2**	**87.2**	81.6	**82.0**	**82.3**	81.7
5	Corral	86.6	**86.9**	98.7	**99.4**	**98.8**	98.6
6	Crx	89.1	**89.2**	84.7	**86.2**	84.1	**84.4**
7	Diabetes	83.5	**83.7**	79.8	**81.8**	79.9	**80.3**
8	Flare	79.9	**80.1**	82.8	**82.9**	81.4	**81.9**
9	German	81.7	**82.7**	**78.8**	78.0	**78.8**	78.4
10	Glass	**79.1**	78.1	**78.1**	76.5	**78.2**	75.7
11	Heart	**87.9**	**87.9**	**82.7**	80.1	**81.0**	80.6
12	Hepatitis	**92.5**	91.1	94.3	**94.5**	**94.0**	92.5
13	Iris	95.8	**95.9**	95.7	**96.5**	**95.1**	94.1
14	Letter	74.2	**74.3**	**87.9**	86.8	**89.2**	88.4
15	Monf-3-7-10	**86.4**	86.0	**91.0**	89.4	91.5	**92.2**
16	Satimage	82.5	**82.6**	88.7	**88.9**	86.8	**87.9**
17	Segment	**92.1**	92.0	**89.2**	85.3	**88.3**	81.8
18	Shuttle	99.3	**99.4**	**99.9**	99.8	**99.9**	99.8
19	Vehicle	**68.6**	**68.6**	76.8	**78.1**	**77.0**	76.0
20	Vote	**90.2**	90.0	93.6	**93.8**	93.6	**94.4**

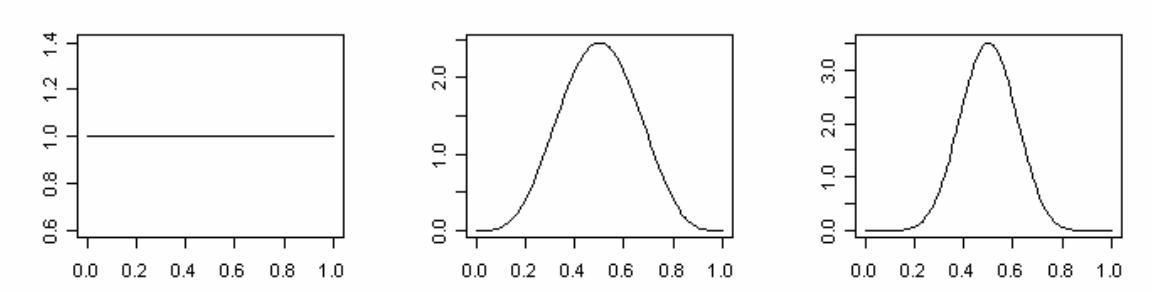

Figura 5.4: Gráfico para un componente en la distribución Dirichlet bidimensional

De izquierda a derecha se considera $\alpha_1 = \alpha_2 = 1$, $\alpha_1 = \alpha_2 = 5$ **y** $\alpha_1 = \alpha_2 = 10$

La tabla 3 muestra la precisión de los clasificadores NB, TAN y CL con y sin suavización de parámetros. Estos resultados muestran que la suavización puede mejorar la exactitud de estos clasificadores en algunos conjuntos de datos. Sin embargo esta mejora no resulta ser significativa en el caso del clasificador NB tal como puede observarse en la figura 5.5.

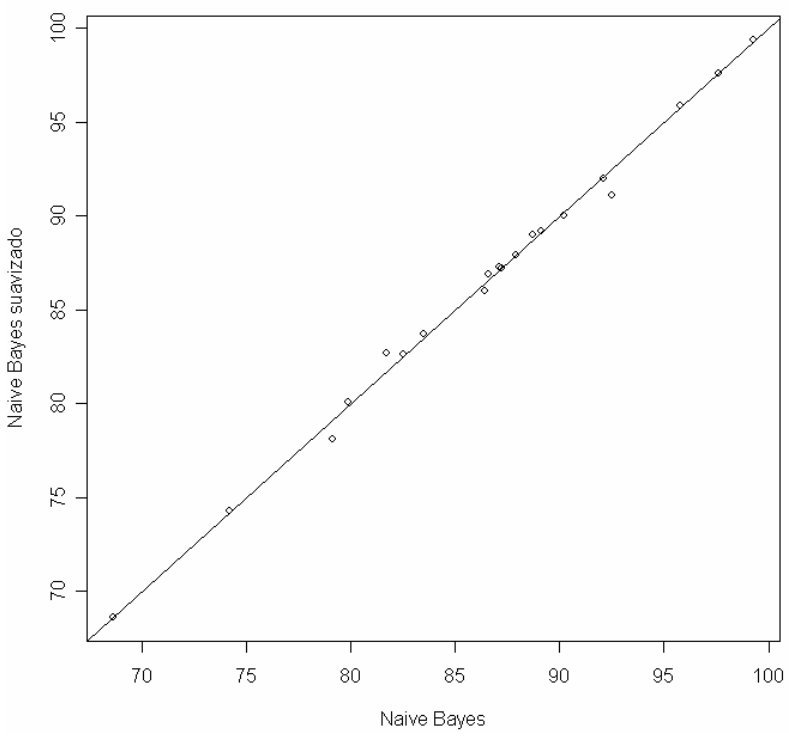

Figura 5.5: Diagramas de dispersión para la precisión obtenida con los clasificadores Naive Bayes y Naive Bayes suavizado

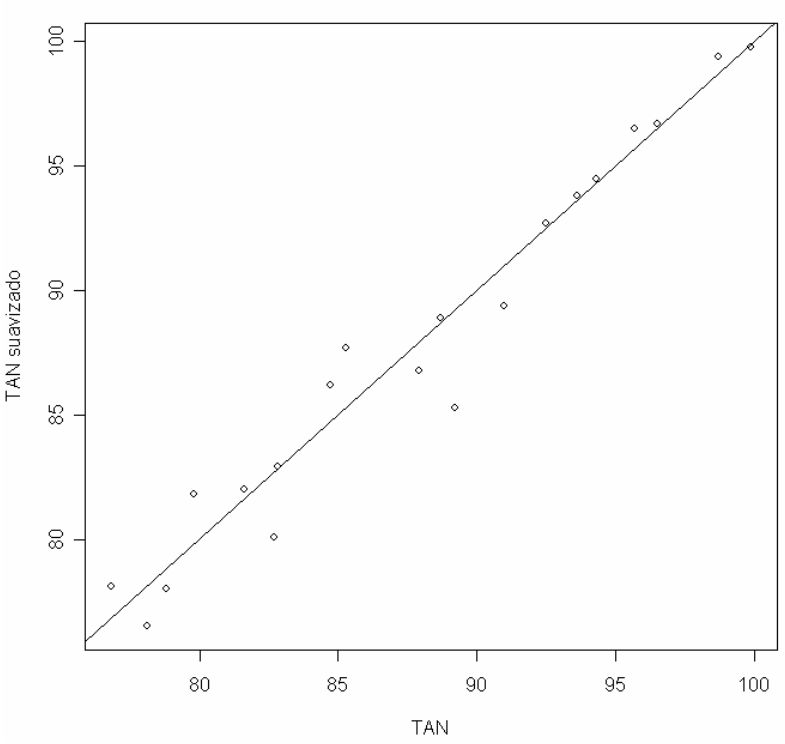

Figura 5.6: Diagramas de dispersión para la precisión obtenida con los clasificadores TAN y TAN suavizado

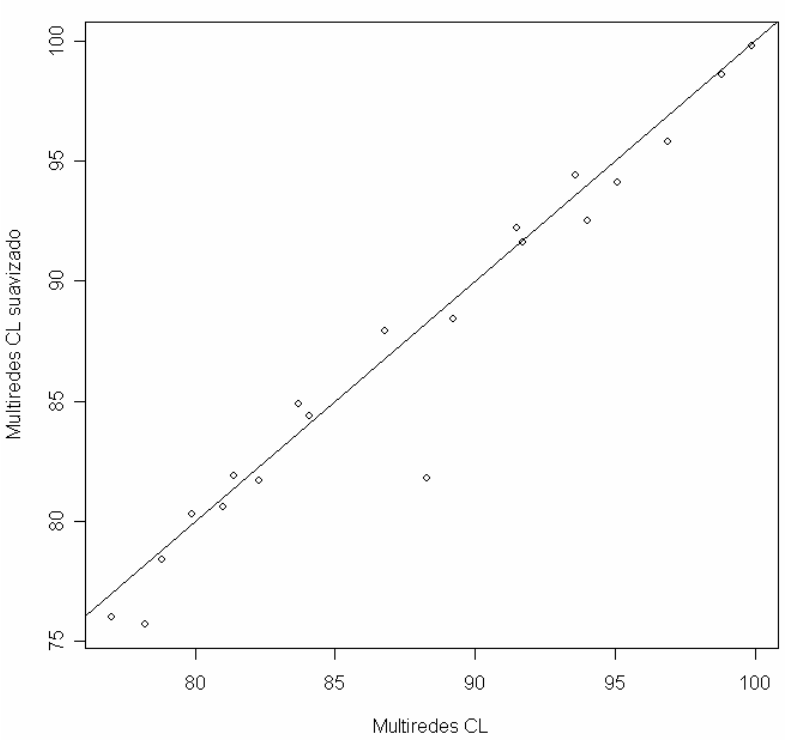

Figura 5.7: Diagramas de dispersión para la precisión obtenida con los clasificadores Multiredes CL y Multiredes CL suavizado

6 CONCLUSIONES Y TRABAJO FUTURO

6.1 Conclusiones

La principal contribución de este trabajo es la evaluación y comparación experimental de los clasificadores Naive Bayes, TAN, y las multiredes Bayesianas CL. Los clasificadores basados en la estructura de dependencia de las variables presentan, en general, un buen comportamiento predictivo. En muchas situaciones resulta útil modelar las relaciones de dependencia entre variables usando una estructura de árbol que nos permita construir un clasificador en base a su estructura. Sin embargo, se recomienda realizar un análisis de las relaciones de dependencia antes de utilizar cualquiera de estos clasificadores.

El esfuerzo computacional requerido por el algoritmo TAN y las multiredes Bayesianas es mayor al requerido por el clasificador Naive Bayes. Las redes TAN y las multiredes Bayesianas requieren la estimación de la estructura de dependencias en la red Bayesiana. Además, el proceso de clasificación utiliza tablas condicionales para cada variable y cada configuración de su padre.

Una restricción con respecto al clasificador TAN y las multiredes Bayesianas esta relacionada con el tamaño del conjunto de datos. Estos modelos suponen que existen suficientes instancias en el conjunto de datos de entrenamiento que permitan estimar

robustamente las probabilidades condicionales. Una alternativa se encuentra en la suavización de parámetros que permite una estimación más confiable para las configuraciones que no se encuentran bien representadas en el conjunto de datos de entrenamiento. En algunos conjuntos de datos es posible aumentar la precisión del clasificador usando este procedimiento. Sin embargo el aumento no resulta ser significativo.

6.2 Trabajo Futuro

Los clasificadores Naive Bayes, TAN y las multiredes Bayesianas CL requieren una prediscretización de las variables continuas para llevar acabo el proceso de estimación de parámetros. Sin embargo, esto puede llevarnos a una inevitable pérdida de información. Es importante considerar algunos métodos que permiten trabajar con ambos tipos de variables.

Bøttcher, S. G. [4] presenta un método de estimación para redes Bayesianas con variables discretas y continuas. El conjunto de nodos en la red es $\mathrm{V} = \Delta \cup \Gamma$, donde Δ y Γ representan el conjunto de nodos discretos y continuos respectivamente. Las correspondientes variables aleatorias se denotan por $\mathbf{X} = (I, Y) = \left((I_\delta)_{\delta \in \Delta}, (Y_\gamma)_{\gamma \in \Gamma} \right)$, donde I e Y corresponden al conjunto de variables discretas y continuas respectivamente. Además no se permite que las variables discretas tengan variables

continuas como padres. La función de probabilidad conjunta puede factorizarse como

$$p(\mathbf{X}) = p(i, y) = \prod_{\delta \in \Delta} p\left(i_\delta / i_{\mathrm{Pa}(\delta)}\right) \prod_{\gamma \in \Gamma} p\left(y_\gamma / i_{\mathrm{Pa}(\gamma)}, y_{\mathrm{Pa}(\gamma)}\right)$$

donde $i_{\mathrm{Pa}(\gamma)}$ y $y_{\mathrm{Pa}(\gamma)}$ representan las observaciones de las partes discretas y continuas respectivamente. Es decir la función de probabilidad conjunta puede descomponerse en una parte puramente discreta y una parte mixta.

Se supone que las funciones de probabilidad local corresponden a distribuciones discretas no restringidas con $p\left(i_\delta / i_{\mathrm{Pa}(\delta)}\right) \geq 0$, $\forall \delta \in \Delta$. En la parte mixta de la red se supone que las distribuciones de probabilidad local corresponden a regresiones lineales Gaussianas si los padres son continuos y cuyos parámetros dependen de la configuración de los padres discretos. Sean $\theta_{\gamma / i_{\mathrm{Pa}(\gamma)}} = \left(m_{\gamma / i_{\mathrm{Pa}(\gamma)}}, \beta_{\gamma / i_{\mathrm{Pa}(\gamma)}}, \sigma^2_{\gamma / i_{\mathrm{Pa}(\gamma)}}\right)$ los parámetros de la distribución. Entonces $\left(Y_\gamma / i_{\mathrm{Pa}(\gamma)}, y_{\mathrm{Pa}(\gamma)}, \theta_{\gamma / i_{\mathrm{Pa}(\gamma)}}\right) \sim \mathrm{N}\left(m_{\gamma / i_{\mathrm{Pa}(\gamma)}} + \beta_{\gamma / i_{\mathrm{Pa}(\gamma)}} y_{\mathrm{Pa}(\gamma)}, \sigma^2_{\gamma / i_{\mathrm{Pa}(\gamma)}}\right)$, donde $\beta_{\gamma / i_{\mathrm{Pa}(\gamma)}}$ son los coeficientes de regresión, $m_{\gamma / i_{\mathrm{Pa}(\gamma)}}$ es el intercepto de la regresión y $\sigma^2_{\gamma / i_{\mathrm{Pa}(\gamma)}}$ es la variancia condicional. La elección de la función de distribución para los parámetros se hace de manera conveniente de tal forma que la distribución posterior sea conocida. Además Bøttcher, S. G. [4] presenta el paquete deal, escrito en el software estadístico R que permite estimar este tipo de redes. Sin embargo, no se considera la construcción de un clasificador en base a la estructura de red estimada.

Li [31] propone dos algoritmos, ETAN y CTAN, que imponen pequeñas restricciones al algoritmo TAN para que pueda trabajar con variables continuas y discretas. Las dependencias en la red Bayesiana se modelan según el tipo de variables involucradas. Si K_i denota las variables discretas y X_i las variables continuas. Si una variable discreta tiene como padre a otra variable discreta se utilizan tablas de probabilidad condicional para representar la dependencia, es decir, $p\left(K_2 = k_2 / K_1 = k_1\right) = p_{k_1 k_2}$. La dependencia entre un padre discreto y uno hijo continuo puede representarse usando una mixtura Gaussiana $p\left(X = x / K = k\right) = \sum_{i=1}^{\mathrm{M}} w_{k,i} \mathrm{N}\left(x; \mu_{k,i}, \sigma_{k,i}\right)$.

La dependencia entre un padre e hijo continuos se especifica usando matrices de covariancia inversas. Suponiendo que $X_1, \cdots, X_n$ están distribuidas conjuntamente como

$$p\left(\mathbf{X}\right) = \left(2\pi\right)^{-n/2} \left|\Sigma\right|^{-1/2} e^{-\frac{1}{2}(\mathbf{X}-\mu)' \Sigma^{-1} (\mathbf{X}-\mu)}$$

La probabilidad condicional $p\left(X_2 = x_2 / X_1 = x_1\right)$ puede obtenerse marginalizando y condicionando la función de probabilidad anterior. Representar la probabilidad condicional en el caso de un padre continuo y un hijo discreto es un gran problema. En este caso simplemente se invierte el tipo de relación obteniendo el tipo padre discreto e hijo continuo.

REFERENCIAS

[1] Akaike, H. (1974). A new look at the statistical model identification. *IEEE Transactions on Automatic Control, 19*, 716-723.

[2] Bazell D. and Aha D. (2001). Ensembles of Classifiers for Morphological Galaxy Classification, *Astrophysical Journal, 548*, 219-223.

[3] Bouckaert, R. R. (1994). Properties of Bayesian network learning algorithms. In R. L. de Mantarás and Poole (Eds.), Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence, 102-109. San Francisco, CA.

[4] Bøttcher, S. G. (2004). Learning Bayesian Networks with Mixed Variables, *Artificial Intelligence and Statistics 2001*, Morgan Kaufmann, San Francisco, CA, USA, 149–156.

[5] Buntine, W. L. (1991). Theory refinement on Bayesian networks. In B. D. D'Ambrosio, P. Smets, and P. P. Bonissone (Eds.), *Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligent*, 52-60. Los Angeles, CA: Morgan Kaufmann.

[6] Buntine, W. L. (1994). Operations for learning with graphical models. *Journal of Artificial Intelligence Research, 2*, 159-225.

[7] Cheng J., Bell, D. A., and Liu, W. (1997) An algorithm for Bayesian belief networks construction from data *In proceeding of the Sixth International Workshop on Artificial Intelligent and Statistics, AI&STAT'97*, 83-90.

[8] Cheng J., Bell, D. A., and Liu, W. (1997) Learning belief networks: as information theory based approach. *In proceeding of the Sixth International ACM Conference on Information and Knowledge Management, CIKM'97.*

[9] Chickering, D.M. (1995). Learning Bayesian networks is NP-complete. In D. Fisher & A. Lenz. *Learning from Data*. Springer-Verlag.

[10] Chow, C. and Liu, C. (1968). Approximating Discrete Probability Distributions with Dependence Trees. *IEEE Transactions on Information Theory, vol 14*, 462-467.

[11] Cooper, G. (1990). Computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence, 42*, 393-405.

[12] Cooper, G. and Herskovits, E. (1992). A Bayesian Method for the Induction of Probabilistic Networks from Data. *Machine Learning*, 9, 309-347.

[13] Dawid, A. P. (1992). Application of general propagation algorithm for probabilistic expert systems. *Statistics and Computing*, 2, 25-36.

[14] Domingos, P. and Pazzani, M. (1997). Beyond Independence: Conditions for the Optimality of the Simple Bayesian Classifier. *Machine Learning, 29*, 103-130.

[15] Duda, R. O. & P. E. Hart. (1973). Pattern Classification and Scene Analysis. New York: John Wiley & Sons.

[16] Even, S. (1979). Graph algorithms. Computer Science Press.

[17] Fayyad, U. M. and K. B. Irani (1993). Multi-interval discretization of continuous-valued attributes for classification learning. *In Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence,* 1022–1027. San Francisco, CA: Morgan Kaufmann.

[18] Friedman, N., Geiger, D. and Goldszmidt, M. (1997). Bayesian Network Classifiers. *Machine Learning, 29*,131-163.

[19] Geiger, D. & D. Heckerman (1996). Knowledge representation and inference in similarity networks and Bayesian multinets. *Artificial Intelligence, 82*, 45–74.

[20] Heckerman, D. (1991). Probabilistic Similarity Networks. Cambridge, MA: MIT Press.

[21] Heckerman, D., Geiger, D. and Chickering, D. (1995). Learning Bayesian Networks: The Combination of Knowledge and Statistical Data. *Machine Learning, 20*, 197-243.

[22] Heckerman, D. (1996). A tutorial on learning with Bayesian networks. *Tech. Rep. No. MSR-TR-95-06*. Redmond, WA: Microsoft Research.

[23] Herskovits, E. and Cooper, G. (1990). Kutató: An Entropy Driven System for the Construction of Probabilistic Expert Systems from Databases. *In Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence.*

[24] Hernández Orallo J., Ferri Ramírez, C., Ramírez Quintana J. (2004). Introducción a la minería de datos. Pearson Educación.

[25] Howard, R. A. and Matheson, J. E. (1983). Influence diagrams. In R. A. Howard and J. E. Matheson (Eds.). *Readings on the principles and applications of decision theory,* 719-762. Strategic Decision Group.

[26] John, G. and R. Kohavi (1997). Wrappers for feature subset selection. Artificial Intelligence. Accepted for publication. A preliminary version appears in Proceedings of the Eleventh International Conference on Machine Learning, 1994,

[27] Kerber, R. (1992). Chimerge, Discretizacion of numeric attributes. *Proceedings of the Tenth National Conference on Artificial Intelligence MIT Press* 123-128.

[28] Kim S., Seo H. and Rim H. (2003). Poisson Naive Bayes for Text Classification with Feature Weighting. *International Workshop on Information Retrieval with Asian Languages*.

[29] Lam, W. and Bacchus, F. (1994). Learning Bayesian Belief Networks. An Approach Based on the MDL Principle. *Computational Intelligence, 10*, 269-293.

[30] Lauritzen, S. L., and Spiegelhalter, D. J. (1988). Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society B*, 50(2), 240-265.

[31] Li, X. (2003). Augmented Bayesian Classifiers for Mixed-Mode Data. http://ssli.ee.washington.edu/~lixiao/

[32] Pearl, J. (1988). Probabilistic Reasoning in Intelligent Systems. San Francisco, CA: Morgan Kaufmann.

[33] Rodríguez, C. (2004). A computacional enviroment for data preprocessing in supervised classification. Thesis MS. Universidad de Puerto Rico, Mayagüez.

[34] Rissanen, J. (1989). Stochastic complexity in statistical inquiry. River Edge, NJ: World Scientific

[35] Sahami, M., Dumais, S., Heckerman, D. and Horvitz E. (1998). A Bayesian Approach to Filtering Junk E-Mail. *In Learning for Text Categorization, Papers from the 1998 Workshop. AAAI Technical Report WS-98-05*.

[36] Schwarz, G. (1978). Estimating the dimension of a model. *Annals of Statistics, 6*, 461-464.

[37] Sebe N., Lew M., Cohen I., Gary A. and Huang T. (2002). Emotion Recognition Using Cauchy Naive Bayes Classifier. *International Conference on Pattern Recognition, Volume 1.*

[38] Singh M. and Voltara, M. (1995). Construction of Bayesian network structures from data: A brief survey and an efficient algorithm. *International Journal of Approximate Reasoning, 12*, 111-113.

[39] Spiegelhalter, D. J., and Lauritzen, S. L. (1990). Sequential updating of conditional probabilities on directed graph structures. *Networks*, 20, 579-605.

[40] Spites, P., Glymour, C., and Scheines, R. (1991). An algorithm for fast recovery of sparse causal graphs. *Social Science Computer Review, 9*, 62-72.

[41] Suzuki, J. (1996). Learning Bayesian Belief Networks Based on the Minimum Description Length Principle: An Efficient Algorithm Using the B&B Technique. *In Proceedings of the Thirteenth International Conference on Machine Learning,* 462-470.

APÉNDICE

Las funciones creadas en el programa estadístico R, que se muestran a continuación, permiten estimar la estructura de una red Bayesiana. Además, construyen y evalúan los diferentes clasificadores mencionados en este trabajo.

MaxSpanTree

```
function(net,p)
{
 net1=net
 sets=matrix(0,p,p)
 finalnet=matrix(0,(p-1),2)
 sets[1,1]=net1[1,1]
 sets[1,2]=net1[1,2]
 finalnet[1,]=as.numeric(net1[1,])
 u=unique(sets[1,])
 u=u[u!=0]
 m=2
 i=2
 while (m<p)
   {
    nodes=as.numeric(net1[i,])
    nsets=as.numeric(table(match(sets[,1],0,nomatch=0))[1])
    matches=matrix(0,nsets,2)
    for (j in 1:nsets)
       {
        matches[j,]=match(nodes,sets[j,],nomatch=0)
       }
    for (l in 1:dim(matches)[1])
       {
        for (h in 1:dim(matches)[2])
           {
            if (matches[l,h]!=0)
              {
               matches[l,h]=1
              }
           }
```

```
   }
inter=apply(matches,1,sum)
if (sum(matches)==0)
 {
   c=which(sets[,1]==0)[1]
   sets[c,1]=nodes[1]
   sets[c,2]=nodes[2]
   finalnet[m,]=as.numeric(nodes)
   i=i+1
   m=m+1
 }
if (sum(matches)==1)
 {
   a=which(inter==1)
   u=unique(sets[a,])
   u=u[u!=0]
   u=union(u,nodes)
   b=length(u)
   for (k in 1:b)
     {
       sets[a,k]=u[k]
     }
   finalnet[m,]=nodes
   i=i+1
   m=m+1
 }
if (sum(matches)==2)
 {
   d=length(which(inter==2))
   if (d==1)
     {
       i=i+1
     }
   if (d==0)
     {
       e=which(inter==1)
       e1=e[1]
       e2=e[2]
       u1=unique(sets[e1,])
       u1=u1[u1!=0]
       u2=unique(sets[e2,])
       u2=u2[u2!=0]
       u=union(u1,u2)
```

```
            b=length(u)
            for (k in 1:b)
               {
                 sets[e1,k]=u[k]
               }
            sets=sets[-e2,]
            finalnet[m,]=nodes
            i=i+1
            m=m+1
          }
        }
   }
 finalnet=finalnet
}
```

MutualInf

```
function(pxy,px,py)
{
 mutinf=0
 for (k in 1:length(px))
    {
     for (l in 1:length(py))
        {
          if (pxy[k,l]!=0)
            {
              mutinf=mutinf+pxy[k,l]*log(pxy[k,l]/(px[k]*py[l]))
            }
        }
    }
 mutinf=mutinf
}
```

BayesNet

```
function(finalnet,p)
{
 dadsonmatrix=matrix(0,p-1,p-1)
 final=finalnet
 root=final[1,1]
 firstdad=final[1,2]
```

```
dad=final[1,2]
matches=matrix(0,dim(finalnet)[1],2)
for (i in 2:dim(matches)[1])
  {
   matches[i,]=match(dad,final[i,],nomatch=0)
  }
a=which(apply(matches,1,sum)!=0)
if (length(a)==0)
 {
  root=final[1,2]
  firstdad=final[1,1]
  dad=final[1,1]
  for (i in 2:dim(matches)[1])
     {
      matches[i,]=match(dad,final[i,],nomatch=0)
     }
  a=which(apply(matches,1,sum)!=0)
 }
b=dad
for (i in 1:length(a))
  {
   b=union(b,final[a[i],])
  }
c=b[b!=dad]
dadsonmatrix[dad,(1:length(c))]=c
final=final[-a,]
if (length(final)==2)
 {
  final=t(as.matrix(final))
 }
final=final[-1,]
dad=c
if (length(final)==2)
 {
  final=t(as.matrix(final))
 }
while (dim(final)[1]!=0)
 {
  d=0
  for (i in 1:length(dad))
     {
      if (dim(final)[1]!=0)
        {
```

```
        matches=matrix(0,length(final)/2,2)
        for (j in 1:dim(matches)[1])
          {
           matches[j,]=match(dad[i],final[j,],nomatch=0)
          }
        a=which(apply(matches,1,sum)!=0)
        if (length(a)!=0)
         {
          b=dad[i]
          for (j in 1:length(a))
             {
               b=union(b,final[a[j],])
             }
          c=b[b!=dad[i]]
          dadsonmatrix[dad[i],(1:length(c))]=c
          final=final[-a,]
          if (length(final)==2)
            {
              final=t(as.matrix(final))
            }
          d=union(d,c)
         }

        if (length(dad)==1 && length(a)==0)
         {
          d=union(d,root)
         }
        if (d==0 && length(a)==0)
         {
          d=union(d,root)
         }
      }
    }
  dad=d[d!=0]
 }
depmatrix=data.frame(matrix(0,(p-1),2))
names(depmatrix)=c("Node 1","Node 2")
depmatrix[,1]=seq(1:(p-1))
for (i in 1:dim(dadsonmatrix)[1])
  {
   if (i==root)
     {
      depmatrix[i,1]=0
```

```
        depmatrix[i,2]=root
       }
      else
       {
        if (i==firstdad)
         {
          depmatrix[i,2]=root
         }
        else
         {
          a=which(dadsonmatrix==i)%%(p-1)
          if (a==0)
           {
            depmatrix[i,2]=p-1
           }
          else
           {
            depmatrix[i,2]=a
           }
         }
       }
    }
 depmatrix=depmatrix
}
```

CrossValNB Validación cruzada Naive Bayes

```
function(data,nparts=10,repet=10)
{
 n=dim(data)[1]
 p=dim(data)[2]
 errorcv=rep(0,repet)
 for (i in 1:repet)
    {
     cat("Repeticion",i,"\n")
     salida=matrix(0,1,nparts)
     azar=data[rank(runif(n)),]
     parti=floor(n/nparts)
     for (j in 1:nparts)
        {
         cc=((j-1)*parti+1):(j*parti)
         if (j==nparts)
```

```
        {
          cc=((j-1)*parti+1):n
        }
        datap=azar[cc,]
        datat=azar[-cc,]
        salida[j]=NB(datat,datap,p)
      }
    errorcv[i]=sum(salida)/n
  }
errorp=mean(errorcv)
sdp=sd(errorcv)
print(errorp)
print(sdp)
}
```

CrossValTAN Validación cruzada TAN

```
function(data,nparts=10,repet=10)
{
 n=dim(data)[1]
 p=dim(data)[2]
 errorcv=rep(0,repet)
 for (i in 1:repet)
   {
    cat("Repeticion",i,"\n")
    salida=matrix(0,1,nparts)
    azar=data[rank(runif(n)),]
    parti=floor(n/nparts)
    for (j in 1:nparts)
      {
       cc=((j-1)*parti+1):(j*parti)
       if (j==nparts)
         {
           cc=((j-1)*parti+1):n
         }
       datap=azar[cc,]
       datat=azar[-cc,]
       salida[j]=TAN(datat,datap,p)
      }
    errorcv[i]=sum(salida)/n
  }
 errorp=mean(errorcv)
```

```
 sdp=sd(errorcv)
 print(errorp)
 print(sdp)
}
```

CrossValCL Validación cruzada Multiredes Bayesianas CL

```
function(data,nparts=10,repet=10)
{
 n=dim(data)[1]
 p=dim(data)[2]
 errorcv=rep(0,repet)
 for (i in 1:repet)
    {
     cat("Repeticion",i,"\n")
     salida=matrix(0,1,nparts)
     azar=data[rank(runif(n)),]
     parti=floor(n/nparts)
     for (j in 1:nparts)
        {
         cc=((j-1)*parti+1):(j*parti)
         if (j==nparts)
           {
             cc=((j-1)*parti+1):n
           }
         datap=azar[cc,]
         datat=azar[-cc,]
         salida[j]=CL(datat,datap,p)
        }
     errorcv[i]=sum(salida)/n
    }
 errorp=mean(errorcv)
 sdp=sd(errorcv)
 print(errorp)
 print(sdp)
}
```

CrossValSNB Validación cruzada Naive Bayes con suavización de parámetros

```
function(data,nparts=10,repet=10,s=5)
{
```

```
n=dim(data)[1]
p=dim(data)[2]
errorcv=rep(0,repet)
for (i in 1:repet)
  {
   cat("Repeticion",i,"\n")
   salida=matrix(0,1,nparts)
   azar=data[rank(runif(n)),]
   parti=floor(n/nparts)
   for (j in 1:nparts)
     {
      cc=((j-1)*parti+1):(j*parti)
      if (j==nparts)
        {
          cc=((j-1)*parti+1):n
        }
      datap=azar[cc,]
      datat=azar[-cc,]
      salida[j]=SNB(datat,datap,p,s)
     }
   errorcv[i]=sum(salida)/n
  }
errorp=mean(errorcv)
sdp=sd(errorcv)
print(errorp)
print(sdp)
}
```

CrossValSTAN Validación cruzada TAN con suavización de parámetros

```
function(data,nparts=10,repet=10,s=5)
{
 n=dim(data)[1]
 p=dim(data)[2]
 errorcv=rep(0,repet)
 for (i in 1:repet)
   {
    cat("Repeticion",i,"\n")
    salida=matrix(0,1,nparts)
    azar=data[rank(runif(n)),]
    parti=floor(n/nparts)
    for (j in 1:nparts)
```

```
      {
       cc=((j-1)*parti+1):(j*parti)
       if (j==nparts)
         {
           cc=((j-1)*parti+1):n
         }
       datap=azar[cc,]
       datat=azar[-cc,]
       salida[j]=STAN(datat,datap,p,s)
      }
    errorcv[i]=sum(salida)/n
  }
errorp=mean(errorcv)
sdp=sd(errorcv)
print(errorp)
print(sdp)
}
```

CrossValSCL Validación cruzada Multiredes Bayesianas CL con suavización de parámetros

```
function(data,nparts=10,repet=10,s=5)
{
 n=dim(data)[1]
 p=dim(data)[2]
 errorcv=rep(0,repet)
 for (i in 1:repet)
   {
    cat("Repeticion",i,"\n")
    salida=matrix(0,1,nparts)
    azar=data[rank(runif(n)),]
    parti=floor(n/nparts)
    for (j in 1:nparts)
      {
       cc=((j-1)*parti+1):(j*parti)
       if (j==nparts)
         {
           cc=((j-1)*parti+1):n
         }
       datap=azar[cc,]
       datat=azar[-cc,]
```

```
        salida[j]=SCL(datat,datap,p,s)
       }
     errorcv[i]=sum(salida)/n
   }
 errorp=mean(errorcv)
 sdp=sd(errorcv)
 print(errorp)
 print(sdp)
}
```

NaiveBayes Naive Bayes data de prueba y entrenamiento

```
function(datat,datap)
{
 p=dim(datat)[2]
 class=sort(unique(datat[,p]))
 cat("Numero de clases",length(class),"\n")
 prior=table(datat[,p])/dim(datat)[1]
 probability=matrix(0,dim(datap)[1],length(class))
 for (k in class)
   {
    cat("Clase",k,"\n")
    dataclass=datat[datat[,p]==k,]
    probdataclass=matrix(0.001,dim(datap)[1],(p-1))
    for (j in 1:(p-1))
       {
        prob=table(dataclass[,j])/dim(dataclass)[1]
        classes=sort(unique(dataclass[,j]))
        for (l in 1:length(classes))
           {
            index=which(datap[,j]==classes[l])
            if (length(index)!=0)
              {
                probdataclass[index,j]=prob[l]
              }
           }
       }
    probability[,k]=prior[k]*apply(probdataclass,1,prod)
   }
 classpredict=max.col(probability)
 salida=mean(classpredict!=datap[,p])
 print(salida)
```

```
}
```

TreeAugNetwork TAN data de prueba y entrenamiento

```
function(datat,datap)
{
 p=dim(datat)[2]
 class=sort(unique(datat[,p]))
 cat("Numero de clases",length(class),"\n")
 infmatrix=matrix(0,(p-1),(p-1))
 for (k in class)
   {
    dataclass=datat[datat[,p]==k,]
    infmatrixtemp=matrix(-Inf,(p-1),(p-1))
    for (i in 1:(p-2))
      {
       for (j in (i+1):(p-1))
          {
           pxy=table(dataclass[,i],dataclass[,j])/dim(dataclass)[1]
           px=table(dataclass[,i])/dim(dataclass)[1]
           py=table(dataclass[,j])/dim(dataclass)[1]
           infmatrixtemp[i,j]=MutualInf(pxy=pxy,px=px,py=py)
          }
       }
    infmatrix=infmatrix+infmatrixtemp
   }
 infmatrix=data.frame(infmatrix)
 names(infmatrix)=names(datat)[-p]
 row.names(infmatrix)=names(datat)[-p]
 nodes=data.frame(1:(p-1))
 names(nodes)="Index"
 row.names(nodes)=names(datat)[-p]
 net=data.frame(matrix(0,(p-2)*(p-1)/2,5))
 names(net)=c("Atrib","Atrib","Node 1","Node 2","Information")
 for (k in 1:((p-2)*(p-1)/2))
   {
    row=match(max(infmatrix),as.matrix(infmatrix))%%(p-1)
    column=floor(match(max(infmatrix),as.matrix(infmatrix))/(p-1))+1
    net[k,1]=row.names(infmatrix)[row]
    net[k,2]=names(infmatrix)[column]
    net[k,3]=row
    net[k,4]=column
```

```
    net[k,5]=infmatrix[row,column]
    infmatrix[row,column]=-Inf
  }
finalnet=MaxSpanTree(net[,c(3,4)],p-1)
depmatrix=BayesNet(finalnet,p)
probability=matrix(0,dim(datap)[1],length(class))
prior=table(datat[,p])/dim(datat)[1]
for (k in 1:length(class))
  {
    cat("Clase",k,"\n")
    dataclass=datat[datat[,p]==k,]
    probdataclass=matrix(0.001,dim(datap)[1],(p-1))
    for (m in 1:dim(depmatrix)[1])
      {
       if (depmatrix[m,1]==0)
        {
         b=depmatrix[m,2]
         prob=table(dataclass[,b])/dim(dataclass)[1]
         classes=sort(unique(dataclass[,b]))
         for (l in 1:length(classes))
           {
            index=which(datap[,b]==classes[l])
            if (length(index)!=0)
             {
              probdataclass[index,b]=prob[l]
             }
           }
        }
       else
        {
         a=depmatrix[m,1]
         b=depmatrix[m,2]
         classb=sort(unique(dataclass[,b]))
         for (s in 1:length(classb))
           {
            datacond=dataclass[dataclass[,b]==classb[s],]
            prob=table(datacond[,a])/dim(datacond)[1]
            classa=as.numeric(names(prob))
            for (l in classa)
              {
               index1=which(datap[,a]==l)
               index2=which(datap[,b]==classb[s])
               index=intersect(index1,index2)
```

```
                if (length(index)!=0)
                  {
                    probdataclass[index,a]=prob[names(prob)==l]
                  }
                }
              }
            }
          }
      probability[,k]=prior[k]*apply(probdataclass,1,prod)
    }
 classpredict=max.col(probability)
 salida=mean(classpredict!=datap[,p])
 print(salida)
}
```

CLMultClass Multiredes Bayesianas CL data de prueba y entrenamiento

```
function(datat,datap,p)
{
 p=dim(datat)[2]
 class=sort(unique(datat[,p]))
 cat("Numero de clases",length(class),"\n")
 prior=table(datat[,p])/dim(datat)[1]
 probability=matrix(0,dim(datap)[1],length(class))
 for (k in class)
    {
     cat("Clase",k,"\n")
     dataclass=datat[datat[,p]==k,]
     infmatrix=matrix(-Inf,(p-1),(p-1))
     for (i in 1:(p-2))
        {
         for (j in (i+1):(p-1))
            {
              pxy=table(dataclass[,i],dataclass[,j])/dim(dataclass)[1]
              px=table(dataclass[,i])/dim(dataclass)[1]
              py=table(dataclass[,j])/dim(dataclass)[1]
              infmatrix[i,j]=MutualInf(pxy=pxy,px=px,py=py)
            }
        }
     infmatrix=data.frame(infmatrix)
     names(infmatrix)=names(datat)[-p]
     row.names(infmatrix)=names(datat)[-p]
```

```
nodes=data.frame(1:(p-1))
names(nodes)="Index"
row.names(nodes)=names(datat)[-p]
net=data.frame(matrix(0,(p-2)*(p-1)/2,5))
names(net)=c("Atrib","Atrib","Node 1","Node 2","Information")
for (i in 1:((p-2)*(p-1)/2))
  {
   row=match(max(infmatrix),as.matrix(infmatrix))%%(p-1)
   column=floor(match(max(infmatrix),as.matrix(infmatrix))/(p-1))+1
   net[i,1]=row.names(infmatrix)[row]
   net[i,2]=names(infmatrix)[column]
   net[i,3]=row
   net[i,4]=column
   net[i,5]=infmatrix[row,column]
   infmatrix[row,column]=-Inf
  }
finalnet=MaxSpanTree(net[,c(3,4)],p-1)
depmatrix=BayesNet(finalnet,p)
probdataclass=matrix(0.001,dim(datap)[1],(p-1))
for (m in 1:dim(depmatrix)[1])
  {
   if (depmatrix[m,1]==0)
    {
     b=depmatrix[m,2]
     prob=table(dataclass[,b])/dim(dataclass)[1]
     classes=sort(unique(dataclass[,b]))
     for (l in 1:length(classes))
       {
        index=which(datap[,b]==classes[l])
        if (length(index)!=0)
          {
           probdataclass[index,b]=prob[l]
          }
       }
    }
   else
    {
     a=depmatrix[m,1]
     b=depmatrix[m,2]
     classb=sort(unique(dataclass[,b]))
     for (s in 1:length(classb))
       {
        datacond=dataclass[dataclass[,b]==classb[s],]
```

```
                prob=table(datacond[,a])/dim(datacond)[1]
                classa=as.numeric(names(prob))
                for (l in classa)
                  {
                   index1=which(datap[,a]==l)
                   index2=which(datap[,b]==classb[s])
                   index=intersect(index1,index2)
                   if (length(index)!=0)
                    {
                     probdataclass[index,a]=prob[names(prob)==l]
                    }
                  }
              }
            }
        }
     probability[,k]=prior[k]*apply(probdataclass,1,prod)
   }
 classpredict=max.col(probability)
 salida=mean(classpredict!=datap[,p])
 print(salida)
}
```

SNaiveBayes Naive Bayes data de prueba y entrenamiento con suavización de parámetros

```
function(datat,datap,s=5)
{
 p=dim(datat)[2]
 class=sort(unique(datat[,p]))
 cat("Numero de clases",length(class),"\n")
 prior=table(datat[,p])/dim(datat)[1]
 probability=matrix(0,dim(datap)[1],length(class))
 for (k in class)
   {
    cat("Clase",k,"\n")
    dataclass=datat[datat[,p]==k,]
    probdataclass=matrix(0.001,dim(datap)[1],(p-1))
    for (j in 1:(p-1))
      {
       Nik=table(datat[,j])
       Nik[seq(1:dim(Nik))]=0
```

```
        N=table(dataclass[,j])
        for (i in 1:length(N))
           {
            Nik[names(N)[[i]]]=N[i]
           }
        if (s==0)
          {
           alphaik=rep(1,length(Nik))
          }
        else
          {
           alphaik=rep(s/length(Nik),length(Nik))
          }
        postprob=(Nik+alphaik)/(sum(Nik)+sum(alphaik))
        classes=sort(unique(datat[,j]))
        for (l in 1:length(classes))
           {
            index=which(datap[,j]==classes[l])
            if (length(index)!=0)
              {
               probdataclass[index,j]=postprob[l]
              }
           }
       }
     probability[,k]=prior[k]*apply(probdataclass,1,prod)
   }
 classpredict=max.col(probability)
 salida=mean(classpredict!=datap[,p])
 print(salida)
}
```

STreeAugNetwork TAN data de prueba y entrenamiento con suavización de parámetros

```
function(datat,datap,s=5)
{
 p=dim(datat)[2]
 class=sort(unique(datat[,p]))
 cat("Numero de clases",length(class),"\n")
 infmatrix=matrix(0,(p-1),(p-1))
 for (k in class)
```

```
  {
   dataclass=datat[datat[,p]==k,]
   infmatrixtemp=matrix(-Inf,(p-1),(p-1))
   for (i in 1:(p-2))
     {
      for (j in (i+1):(p-1))
        {
         pxy=table(dataclass[,i],dataclass[,j])/dim(dataclass)[1]
         px=table(dataclass[,i])/dim(dataclass)[1]
         py=table(dataclass[,j])/dim(dataclass)[1]
         infmatrixtemp[i,j]=MutualInf(pxy=pxy,px=px,py=py)
        }
     }
   infmatrix=infmatrix+infmatrixtemp
  }
infmatrix=data.frame(infmatrix)
names(infmatrix)=names(datat)[-p]
row.names(infmatrix)=names(datat)[-p]
nodes=data.frame(1:(p-1))
names(nodes)="Index"
row.names(nodes)=names(datat)[-p]
net=data.frame(matrix(0,(p-2)*(p-1)/2,5))
names(net)=c("Atrib","Atrib","Node 1","Node 2","Information")
for (k in 1:((p-2)*(p-1)/2))
  {
   row=match(max(infmatrix),as.matrix(infmatrix))%%(p-1)
   column=floor(match(max(infmatrix),as.matrix(infmatrix))/(p-1))+1
   net[k,1]=row.names(infmatrix)[row]
   net[k,2]=names(infmatrix)[column]
   net[k,3]=row
   net[k,4]=column
   net[k,5]=infmatrix[row,column]
   infmatrix[row,column]=-Inf
  }
finalnet=MaxSpanTree(net[,c(3,4)],p-1)
depmatrix=BayesNet(finalnet,p)
probability=matrix(0,dim(datap)[1],length(class))
prior=table(datat[,p])/dim(datat)[1]
for (k in 1:length(class))
  {
   cat("Clase",k,"\n")
   dataclass=datat[datat[,p]==k,]
   probdataclass=matrix(0.001,dim(datap)[1],(p-1))
```

```
for (m in 1:dim(depmatrix)[1])
  {
   if (depmatrix[m,1]==0)
    {
     b=depmatrix[m,2]
     Nik=table(datat[,b])
     Nik[seq(1:dim(Nik))]=0
     N=table(dataclass[,b])
     for (i in 1:length(N))
       {
        Nik[names(N)[[i]]]=N[i]
       }
     if (s==0)
      {
       alphaik=rep(1,length(Nik))
      }
     else
      {
       alphaik=rep(s/length(Nik),length(Nik))
      }
     postprob=(Nik+alphaik)/(sum(Nik)+sum(alphaik))
     classes=sort(unique(datat[,b]))
     for (l in 1:length(classes))
       {
        index=which(datap[,b]==classes[l])
        if (length(index)!=0)
          {
           probdataclass[index,b]=postprob[l]
          }
       }
    }
   else
    {
     a=depmatrix[m,1]
     b=depmatrix[m,2]
     classb=sort(unique(dataclass[,b]))
     for (r in 1:length(classb))
       {
        datacond=dataclass[dataclass[,b]==classb[r],]
        datacondt=datat[datat[,b]==classb[r],]
        classa=sort(unique(datacond[,a]))
        classat=sort(unique(datacondt[,a]))
        Nijk=table(datacondt[,a])
```

```
                Nijk[seq(1:dim(Nijk))]=0
                N=table(datacond[,a])
                for (i in 1:length(N))
                   {
                     Nijk[names(N)[[i]]]=N[i]
                   }
                if (s==0)
                  {
                    alphaijk=rep(1,length(Nijk))
                  }
                else
                  {
                    alphaijk=rep(s/length(Nijk),length(Nijk))
                  }
                postprob=(Nijk+alphaijk)/(sum(Nijk)+sum(alphaijk))
                for (l in classat)
                   {
                     index1=which(datap[,a]==l)
                     index2=which(datap[,b]==classb[r])
                     index=intersect(index1,index2)
                     if (length(index)!=0)
                       {
                         probdataclass[index,a]=postprob[names(postprob)==l]
                       }
                   }
              }
          }
      }
    probability[,k]=prior[k]*apply(probdataclass,1,prod)
  }

 classpredict=max.col(probability)
 salida=mean(classpredict!=datap[,p])
 print(salida)
}
```

SCLMultClass Multiredes Bayesianas CL data de prueba y entrenamiento con suavización de parámetros

```
function(datat,datap,p,s=5)
{
```

```
p=dim(datat)[2]
class=sort(unique(datat[,p]))
cat("Numero de clases",length(class),"\n")
prior=table(datat[,p])/dim(datat)[1]
probability=matrix(0,dim(datap)[1],length(class))
for (k in class)
  {
   cat("Clase",k,"\n")
   dataclass=datat[datat[,p]==k,]
   infmatrix=matrix(-Inf,(p-1),(p-1))
   for (i in 1:(p-2))
     {
      for (j in (i+1):(p-1))
        {
         pxy=table(dataclass[,i],dataclass[,j])/dim(dataclass)[1]
         px=table(dataclass[,i])/dim(dataclass)[1]
         py=table(dataclass[,j])/dim(dataclass)[1]
         infmatrix[i,j]=MutualInf(pxy=pxy,px=px,py=py)
        }
     }
   infmatrix=data.frame(infmatrix)
   names(infmatrix)=names(datat)[-p]
   row.names(infmatrix)=names(datat)[-p]
   nodes=data.frame(1:(p-1))
   names(nodes)="Index"
   row.names(nodes)=names(datat)[-p]
   net=data.frame(matrix(0,(p-2)*(p-1)/2,5))
   names(net)=c("Atrib","Atrib","Node 1","Node 2","Information")
   for (i in 1:((p-2)*(p-1)/2))
     {
      row=match(max(infmatrix),as.matrix(infmatrix))%%(p-1)
      column=floor(match(max(infmatrix),as.matrix(infmatrix))/(p-1))+1
      net[i,1]=row.names(infmatrix)[row]
      net[i,2]=names(infmatrix)[column]
      net[i,3]=row
      net[i,4]=column
      net[i,5]=infmatrix[row,column]
      infmatrix[row,column]=-Inf
     }
   finalnet=MaxSpanTree(net[,c(3,4)],p-1)
   depmatrix=BayesNet(finalnet,p)
   probdataclass=matrix(0.001,dim(datap)[1],(p-1))
   for (m in 1:dim(depmatrix)[1])
```

```
{
 if (depmatrix[m,1]==0)
  {
   b=depmatrix[m,2]
   Nik=table(datat[,b])
   Nik[seq(1:dim(Nik))]=0
   N=table(dataclass[,b])
   for (i in 1:length(N))
     {
      Nik[names(N)[[i]]]=N[i]
     }
   if (s==0)
    {
     alphaik=rep(1,length(Nik))
    }
   else
    {
     alphaik=rep(s/length(Nik),length(Nik))
    }
   postprob=(Nik+alphaik)/(sum(Nik)+sum(alphaik))
   classes=sort(unique(datat[,b]))
   for (l in 1:length(classes))
     {
      index=which(datap[,b]==classes[l])
      if (length(index)!=0)
        {
         probdataclass[index,b]=postprob[l]
        }
     }
  }
 else
  {
   a=depmatrix[m,1]
   b=depmatrix[m,2]
   classb=sort(unique(dataclass[,b]))
   for (r in 1:length(classb))
     {
      datacond=dataclass[dataclass[,b]==classb[r],]
      datacondt=datat[datat[,b]==classb[r],]
      classa=sort(unique(datacond[,a]))
      classat=sort(unique(datacondt[,a]))
      Nijk=table(datacondt[,a])
      Nijk[seq(1:dim(Nijk))]=0
```

```
                N=table(datacond[,a])
                for (i in 1:length(N))
                  {
                   Nijk[names(N)[[i]]]=N[i]
                  }
                if (s==0)
                 {
                  alphaijk=rep(1,length(Nijk))
                 }
                else
                 {
                  alphaijk=rep(s/length(Nijk),length(Nijk))
                 }
                postprob=(Nijk+alphaijk)/(sum(Nijk)+sum(alphaijk))
                for (l in classat)
                  {
                   index1=which(datap[,a]==l)
                   index2=which(datap[,b]==classb[r])
                   index=intersect(index1,index2)
                   if (length(index)!=0)
                     {
                      probdataclass[index,a]=postprob[names(postprob)==l]
                     }
                  }
               }
            }
        }
     probability[,k]=prior[k]*apply(probdataclass,1,prod)
   }
 classpredict=max.col(probability)
 salida=mean(classpredict!=datap[,p])
 print(salida)
}
```