Universidad Politécnica
de Madrid

Escuela Técnica Superior de Ingenieros Informáticos

Máster Universitario en Inteligencia Artificial

Trabajo Fin de Máster

Cadena de clasificadores bayesianos en tiempo continuo

Autor(a): Santiago González-Carvajal Centenera
Tutor(a): Concha Bielza y Pedro Larrañaga

Madrid, julio - 2021

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

Trabajo Fin de Máster
Máster Universitario en Inteligencia Artificial

Título: Cadena de clasificadores bayesianos en tiempo continuo

julio - 2021

Autor(a): Santiago González-Carvajal Centenera
Tutor(es): Concha Bielza y Pedro Larrañaga
Departamento de inteligencia artificial
ETSI Informáticos
Universidad Politécnica de Madrid

Resumen

El trabajo presenta un nuevo modelo para clasificación multidimensional de series multivariadas. Específicamente, presenta una cadena de clasificadores bayesianos en tiempo continuo.

Con ese objetivo en mente, primero se sientan las bases del trabajo mediante el estudio del estado del arte para problemas de clasificación tanto multidimensionales como temporales. A continuación, se define el nuevo modelo de clasificación de manera formal, y se explica cómo se ha realizado la implementación del mismo. Finalmente, se realizan tres experimentos que muestran que este nuevo modelo tiene sentido, y que este tipo de cadenas de clasificadores en tiempo continuo presentan algunas de las propiedades de las cadenas de clasificadores en el caso estático.

Abstract

This work introduces a new model for multidimensional classification of multivariate time series. Specifically, it introduces a continuous time Bayesian chain classifier.

With that goal in mind, the foundations of the work are first laid by studying the state of the art for both multidimensional and temporal classification problems. Next, the new classification model is formally defined, and how it has been implemented is explained. Finally, three experiments are carried out in order to show that this new model makes sense, and that this type of continuous time chain classifiers presents some of the properties of classifier chains in the static case.

Tabla de contenidos

1. Introducción **1**
1.1. Clasificación multidimensional 1
1.2. Clasificación de series temporales 1
1.3. Objetivos 2
1.4. Notación 3

2. Estado del arte **5**
2.1. Clasificadores multidimensionales 5
2.2. Clasificadores bayesianos multidimensionales 6
2.3. Clasificadores para problemas temporales 8
2.3.1. Clasificadores en tiempo discreto 8
2.3.2. Clasificadores en tiempo continuo 8

3. Definición del modelo **11**
3.1. Clasificadores bayesianos en tiempo continuo 11
3.1.1. Redes bayesianas en tiempo continuo 11
3.1.2. Clasificador bayesiano en tiempo continuo 13
3.1.3. Aprendizaje de un CTBNC 14
3.1.4. Clasificación 16
3.2. Clasificadores en cadena 17
3.3. Cadena de clasificadores bayesianos en tiempo continuo 19

4. Implementación **21**

5. Experimentos **25**
5.1. Experimento 1: Experimento de referencia 26
5.1.1. Descripción del experimento 26
5.1.2. Resultados del experimento 27
5.2. Experimento 2: Clasificadores perfectos 28
5.2.1. Descripción del experimento 28
5.2.2. Resultados del experimento 28
5.3. Experimento 3: Orden de las variables en la cadena 29
5.3.1. Descripción del experimento 29
5.3.2. Resultados del experimento 29
5.4. Conclusiones 30

6. Conclusiones y trabajo futuro **31**

Bibliografía **34**

Anexo **35**

Capítulo 1

Introducción

Hoy en día, y cada vez más, se generan enormes cantidades de datos: con cada acción que realizamos al navegar por Internet, cuando asistimos a una consulta médica, al comprar con tarjeta en un supermercado, etc. Estos datos contienen una cantidad enorme de información acerca de nosotros y pueden ser explotados de muchas maneras diferentes según el objetivo que se persiga.

Los modelos de aprendizaje automático tratan, a grandes rasgos, de encontrar patrones en estos datos que permitan, en el caso de modelos predictivos, llegar a predecir comportamientos futuros. Por lo tanto, los modelos de aprendizaje automático, al igual que estos datos, tienen la necesidad de ir evolucionando con el paso del tiempo para ajustarse al tipo de datos que se generan.

1.1. Clasificación multidimensional

El problema de clasificación tradicional es un problema unidimensional en el cual a partir de un conjunto de variables, $F = \{X_1, \ldots, X_m\}$, se asigna el valor de una variable clase a cada una de las muestras, $y \in Y = \{y_1, \ldots, y_k\}$. El caso particular $k = 2$ se denomina clasificación binaria.

En el caso de los clasificadores multidimensionales, cada muestra puede pertenecer a un conjunto de clases (en vez de únicamente a una clase). Es decir, dado un conjunto de variables, $F = \{X_1, \ldots, X_m\}$, se asigna un valor que representa el conjunto de clases $\mathbf{y} \in \mathbf{Y} = (Y_1, \ldots, Y_d)$ a cada una de las muestras, donde cada $Y_i = \{y_{i_1}, \ldots, y_{i_{p_i}}\}$ con $i = 1, \ldots, d$. Cada p_i con $i = 1, \ldots, d$ representa la cardinalidad del conjunto de etiquetas disponible para cada dimensión.

Nótese que estos p_i no tienen por qué tener el mismo valor. Pero en el caso de que $p_i = 2$ para todo i, el problema se denomina de clasificación multietiqueta, que no es otra cosa que un problema de clasificación binaria en cada una de las dimensiones.

1.2. Clasificación de series temporales

Como veíamos anteriormente, tradicionalmente existe un conjunto de variables, $F = \{X_1, \ldots, X_m\}$, a partir del cual se asigna un conjunto de clases. En estos casos, cada

muestra se analiza de manera "independiente" del resto en el sentido de que el conjunto de clases que se le asigna depende únicamente del conjunto de variables que presenta.

No obstante, cuando aparece una variable que evoluciona con el tiempo, t, si se toman muestras del conjunto de variables en instantes determinados de tiempo, $F(t) = \{X_1(t), \ldots, X_m(t)\}$, la asignación del conjunto de clases puede depender del conjunto de variables en el instante previo e incluso del conjunto de clases previo. Y, por consiguiente, es necesario que el modelo tenga en cuenta esta dependencia temporal.

En ese sentido, la clasificación tradicional unidimensional de series temporales univariadas consiste en asignar una clase $y(t) \in Y = \{y_1, \ldots, y_k\}$ de manera continua a lo largo del tiempo, t, a una trayectoria que representa la evolución de una variable a lo largo del tiempo, $x(t)$. En cambio, cuando hablamos de clasificación multidimensional de series temporales multivariadas, estamos hablando de asignar un conjunto de clases (multidimensional), $\mathbf{y}(t) \in \mathbf{Y} = (Y_1, \ldots, Y_d)$, de manera continua en el tiempo, t, a un conjunto de trayectorias que representan la evolución de un conjunto de variables a lo largo del tiempo (serie temporal multivariada), $\mathbf{x}(t) = \{x_1(t), \ldots, x_m(t)\}$. Y en ambos casos, tal y como se ha comentado, existen dependencias temporales para la clasificación, y es por eso que una de las aproximaciones posibles es utilizar un modelo capaz de tenerlas en cuenta.

1.3. Objetivos

Este trabajo se va a centrar en el análisis de datos temporales. Concretamente, en el problema de clasificación multidimensional de series temporales multivariadas. Para ello, se va a seguir una aproximación basada en clasificadores bayesianos. Específicamente, se van a utilizar los modelos de clasificación supervisada para series temporales presentados en Stella y Amer (2012): clasificadores bayesianos en tiempo continuo, que de ahora en adelante serán abreviados utilizando su acrónimo procedente del inglés, CTBNCs.

No obstante, dado que los CTBNCs solo está preparado para clasificar series temporales multivariadas con una etiqueta, estos serán adaptado para clasificar en múltiples etiquetas siguiendo el enfoque de utilizar clasificadores bayesianos en cadena (Zaragoza *et al.*, 2011). Esta adaptación de los CTBNCs será denominada de ahora en adelante cadena de clasificadores bayesianos en tiempo continuo, y será abreviada en el resto de este documento como CTBNCCs.

Dado que en Stella y Amer (2012) y Codecasa y Stella (2014b) se presentan distintos tipos de clasificadores pertenecientes a la clase de los CTBNCs, al igual que en el caso de clasificadores estático (Bielza y Larrañaga, 2014), los experimentos se llevarán a cabo variando el tipo de clasificador base de la cadena entre ellos, analizando de esta manera, también, cómo varían los resultados en función de las restricciones aplicadas al clasificador base.

De esta manera, los objetivos principales de este proyecto son los siguientes:

- Desarrollar una cadena de clasificadores bayesianos en tiempo continuo y una relevancia binaria para clasificadores bayesianos en tiempo continuo.
- Mostrar que las cadenas de clasificadores bayesianos tienen sentido en tiempo

continuo. Para ello, este tipo de clasificadores será comparado con la relevancia binaria para clasificadores bayesianos en tiempo continuo, utilizando en ambos casos el mismo clasificador base.

- Mostrar que, realmente, la cadena de clasificadores en tiempo continuo es capaz de capturar las relaciones entre variables clase, y que, por consiguiente, el orden a la hora de predecir las variables clase en la cadena influye en el resultado.

La organización de este documento es la siguiente. En el capítulo 2 se presentará el estado del arte en cuanto a clasificación bayesiana se refiere. En el capítulo 3 se explicará en profundidad el diseño de los modelos CTBNCCs, haciendo especial hincapié en los CTBNCs que se van a utilizar como clasificadores base. En el capítulo 4 se darán algunos detalles sobre la implementación del mismo. En el capítulo 5 se presentarán los experimentos realizados y sus resultados. Y, finalmente, en el capítulo 6 se terminará presentando las conclusiones extraídas del trabajo y algunas posibles líneas de trabajo futuro.

1.4. Notación

En este apartado presentamos un resumen de la notación que utilizaremos a lo largo del documento.

La notación que vamos a utilizar de manera implícita es la siguiente:

x	es un escalar.
$\mathbf{x}$	es un vector.
$\mathbf{X}$	es una matriz.

El resto de notación utilizada será explicada en el momento específico en el que se use para facilitar la lectura del documento.

Capítulo 2

Estado del arte

En este capítulo se va a repasar el estado del arte para clasificación multidimensional y para clasificación en tiempo continuo. Ambos tanto desde un punto de vista general como desde un punto de vista de los clasificadores bayesianos.

2.1. Clasificadores multidimensionales

Los problemas de clasificación multidimensionales, al igual que el caso particular de los problemas de clasificación multietiqueta, cuentan con dos grandes grupos de aproximaciones para su resolución: *métodos que transforman el problema* y *métodos de adaptación de algoritmos* (Gil-Begue *et al.*, 2021).

La idea de los primeros es transformar el problema multidimensional en problemas unidimensionales, mientras que la idea de los segundos es modificar un algoritmo existente para que sea capaz de manejar datos multidimensionales.

Dos enfoques sencillos del primer tipo de aproximación son la relevancia binaria y el paso al conjunto potencia (Boutell *et al.*, 2004). La relevancia binaria consiste en construir un clasificador de manera independiente para cada una de las dimensiones por separado. Es decir, si el problema multidimensional tiene d dimensiones, entonces se construirán d clasificadores unidimensionales independientes. Mientras que el paso al conjunto potencia consiste, en generar una única etiqueta que representa todas las posibles combinaciones de las d etiquetas.

El problema de la relevancia binaria es que al generar clasificadores independientes no es capaz de capturar ningún tipo de relación entre las variables clase. Y el principal problema del paso al conjunto potencia es que se genera un número muy grande de posibles valores para la variable clase y hay configuraciones con muy pocas apariciones en el conjunto de entrenamiento.

Con el objetivo de resolver algunos de estos problemas, se han propuesto algunas modificaciones de los anteriores. Este es el caso, por ejemplo, de las cadenas de clasificadores (Read *et al.*, 2011). La cadena de clasificadores consiste básicamente en que la clase predicha por cada clasificador unidimensional de la cadena, se utiliza como entrada para todos los clasificadores posteriores, superando de esta manera el problema de no ser capaz de modelar las relaciones entre variables clase. La Figura 2.1 muestra la clasificación mediante una cadena de clasificadores.

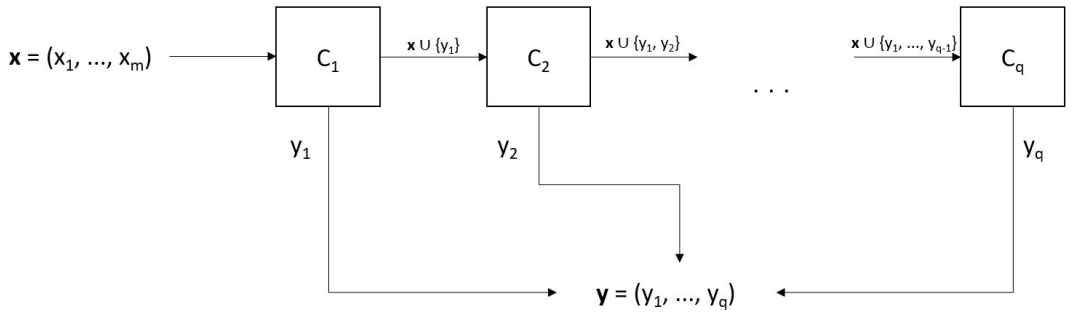

Figura 2.1: Clasificación dada una instancia de las variables predictoras, $\mathbf{x} = (x_1, \ldots, x_m)$, mediante una cadena de clasificadores, $C_1, \ldots, C_q$. El resultado de la clasificación es $\mathbf{y}$, construido a partir de la predicción de cada clasificador base, que a su vez se incorpora al conjunto de variables predictoras del siguiente clasificador. Nótese que el i-ésimo clasificador C_i utiliza como variables predictoras $\mathbf{x}, y_1, \ldots, y_{i-1}$.

No obstante, aparece otro problema: dependiendo del orden en el que se predicen las clases, el resultado varía (ya que dependiendo del orden se pueden capturar distintas dependencias entre variables clase).

Para solucionar este problema, se han presentado distintos métodos. El primero fue el presentado en Read *et al.* (2011), que es construir un conjunto de clasificadores en cadena. Otro posible método sería la cadena de clasificadores circular (Rivas *et al.*, 2018) que consiste en seguir iterando sobre repeticiones de la cadena de clasificadores inicial hasta que las predicciones se estabilizan. Otra posible solución consiste en cadenas de clasificadores dinámicas (Kulessa y Mencía, 2018; Mencía, 2020). Este tipo de aproximación consiste en determinar de manera dinámica el orden de la cadena de clasificadores para cada una de las instancias de test. Y, lógicamente, uno de los problemas que presenta es la carga computacional ya que tiene que utilizar un modelo adicional (metaclasificador) para decidir el orden de las variables para cada instancia.

2.2. Clasificadores bayesianos multidimensionales

En este apartado se va a utilizar C en lugar de Y para referirse a las variables clase ya que en la literatura que se refiere a este tipo clasificadores se utiliza esa notación.

De acuerdo a van der Gaag y de Waal (2006), un clasificador multidimensional bayesiano (MBC) sobre un conjunto de variables aleatorias (v.v.a.a.) discretas $\mathcal{V} = \{Z_1, \ldots, Z_n\}$, con $n \geq 1$, es una red bayesiana $\mathcal{B} = (\mathcal{G}, \Theta)$, donde

- $\mathcal{G}$ es un grafo dirigido acíclico (DAG) cuyos vértices son Z_i, con $i = 1, \ldots, n$. Θ es el conjunto de parámetros: $\theta_{z|\mathbf{pa}(z)} = P(z|\mathbf{pa}(z))$, donde $\mathbf{pa}(z)$ es la instanciación del conjunto de los padres de Z en $\mathcal{G}$, $\mathbf{Pa}(Z)$.

La red bayesiana $\mathcal{B}$ define la distribución de probabilidad conjunta sobre $\mathcal{V}$

$$P_{\mathcal{B}}(z_1, \ldots, z_n) = \prod_{i=1}^{n} P_{\mathcal{B}}(z_i|\mathbf{pa}(z_i))$$

El conjunto de vértices $\mathcal{V}$ se particiona en dos subconjuntos: el de variables clase, $\mathcal{V}_\mathcal{C} = \{C_1, \ldots, C_d\}$, $d \geq 1$, y el de variables predictoras, $\mathcal{V}_\mathcal{X} = \{X_1, \ldots, X_m\}$, $m \geq 1$, y se cumple que $n = d + m$. A su vez, el conjunto de arcos también se particiona en tres subconjuntos:

- $\mathcal{A}_\mathcal{C} \subseteq \mathcal{V}_\mathcal{C} \times \mathcal{V}_\mathcal{C}$ formado por los arcos entre variables clase, y dan lugar al subgrafo clase $\mathcal{G}_\mathcal{C} = (\mathcal{V}_\mathcal{C}, \mathcal{A}_\mathcal{C})$ de $\mathcal{G}$ inducido por $\mathcal{V}_\mathcal{C}$.
- $\mathcal{A}_\mathcal{X} \subseteq \mathcal{V}_\mathcal{X} \times \mathcal{V}_\mathcal{X}$ formado por los arcos entre variables predictoras, y dan lugar al subgrafo atributo $\mathcal{G}_\mathcal{X} = (\mathcal{V}_\mathcal{X}, \mathcal{A}_\mathcal{X})$ de $\mathcal{G}$ inducido por $\mathcal{V}_\mathcal{X}$.
- $\mathcal{A}_{\mathcal{C}\mathcal{X}} \subseteq \mathcal{V}_\mathcal{C} \times \mathcal{V}_\mathcal{X}$ formado por los arcos de variables clase a variables predictoras, y dan lugar al subgrafo puente $\mathcal{G}_{\mathcal{C}\mathcal{X}} = (\mathcal{V}, \mathcal{A}_{\mathcal{C}\mathcal{X}})$ de $\mathcal{G}$ que conecta las variables clase y las predictoras.

La Figura 2.2 muestra un ejemplo.

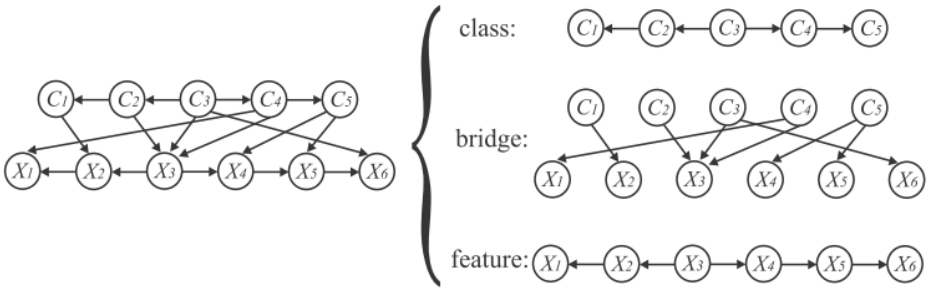

Figura 2.2: MBC con sus tres subgrafos. Fuente Bielza *et al.* (2011).

Este tipo de clasificadores permiten abordar el problema de clasificación multidimensional desde el punto de vista de los clasificadores bayesianos (van der Gaag y de Waal, 2006; Bielza *et al.*, 2011).

Una manera distinta de abordar este problema de clasificación multidimensional es mediante el uso de cadenas de clasificadores tal y como se comentaba en la Sección 2.1. Concretamente, al aplicar la cadena de clasificadores a los clasificadores bayesianos, lo que se obtiene es lo que se presenta en Zaragoza *et al.* (2011): las Cadenas de Clasificadores bayesianos, que básicamente, extienden la idea general de las cadenas de clasificadores utilizando la propiedad de la redes bayesianas para el cálculo de la distribución de probabilidad conjunta. De manera formal, el objetivo es encontrar la distribución conjunta de las clases $\mathbf{C} = (C_1, \ldots, C_d)$ dadas las variables predictoras $\mathbf{x} = (x_1, \ldots, x_m)$:

$$P(\mathbf{C}|\mathbf{x}) = \prod_{i=1}^{d} P(C_i|\mathbf{pa}(C_i), \mathbf{x}),$$

donde $\mathbf{pa}(C_i)$, representa los padres de la clase C_i. Lo ideal es construir la cadena de clasificadores de tal manera que primero aparezcan los clasificadores que predicen clases que no dependen de otras clases, y después continuar con los clasificadores que predicen sus variables hijas. Para ello, se crea un orden parcial de las clases en la cadena de clasificadores que depende de las dependencias entre las clases dadas

las variables predictoras. Asumiendo que estas dependencias se pueden representar mediante una red bayesiana, la estructura de la cadena de clasificadores, es decir, el orden de los clasificadores de la cadena, queda definido por la estructura de la red bayesiana.

2.3. Clasificadores para problemas temporales

En este tipo de clasificadores podemos diferenciar dos grandes grupos: los clasificadores en tiempo discreto y los clasificadores en tiempo continuo.

2.3.1. Clasificadores en tiempo discreto

Los clasificadores que discretizan la componente temporal son, por ejemplo, las redes bayesianas dinámicas (DBNs). Estas son sistemas temporales dinámicos (Dean y Kanazawa, 1989) que representan el tiempo de manera implícita a través de rodajas temporales (Codecasa y Stella, 2014b). No obstante, este tipo de redes presenta un problema: la granularidad temporal, ya que los incrementos temporales han de ser fijos. Otro ejemplo de este tipo de clasificadores que discretizan la variable temporal serían las modelos de redes dinámicas (Dagum *et al.*, 1992) o los conocidos modelos ocultos de Markov (Rabiner, 1989).

También, en Tucker *et al.* (2006) se presentó una extensión de los clasificadores bayesianos llamada clasificador bayesiano temporal, que extiende los clasificadores bayesianos con un nodo raíz, distinto del nodo clase, cuyo estado está asociado con puntos temporales discretos (Stella y Amer, 2012).

No se entrará en más detalles en este tipo de clasificadores ya que los que nos interesan para este trabajo son los clasificadores capaces de representar el tiempo de manera dinámica superando los problemas presentados por los clasificadores anteriores, tales como la granularidad de las DBNs que se comentaba anteriormente.

2.3.2. Clasificadores en tiempo continuo

Este tipo de clasificadores representan el tiempo de manera explícita. Algunos ejemplos de este tipo de clasificadores serían los *noisy-or* en tiempo continuo (Simma *et al.*, 2012), las cascadas de Poisson (Simma, 2010), las redes de Poisson (Rajaram *et al.*, 2005), el modelo de intensidad condicional *piecewise-constant* (Gunawardana *et al.*, 2011), o los procesos puntuales basados en árboles (Weiss y Page, 2013). Todos estos modelos tienen en común que son capaces de analizar procesos en tiempo continuo al ser capaces de representar el tiempo de manera explícita.

Entre estos clasificadores también se encuentran los clasificadores bayesianos en tiempo continuo (con su acrónimo en inglés CTBNC). Fueron introducidos en Stella y Amer (2012) como una nueva clase para clasificación supervisada de series temporales multivariadas. En este capítulo no se entrará en mayor detalle ya que esta clase de clasificadores será explicada en el Capítulo 3. Pero hay que destacar que, tal y como se comentaba anteriormente, este tipo de clasificadores surge para superar el problema de los clasificadores explicados en la Sección 2.3.1 (como el de la granularidad temporal de las DBNs), y representar el tiempo de manera explícita y dinámica (Codecasa y Stella, 2014b).

En ese mismo trabajo (Stella y Amer, 2012), los autores introducen dos clasificadores pertenecientes a esta clase: el clasificador naive Bayes en tiempo continuo (con su acrónimo en inglés CTNB) y el clasificador naive Bayes aumentado en árbol en tiempo continuo (con su acrónimo en inglés CTTANB). Estos dos modelos, además de las restricciones temporales que incorporan, son básicamente las versiones en tiempo continuo de los clasificadores estáticos naive Bayes (con su acrónimo en inglés NB) y naive Bayes aumentado en árbol (con su acrónimo en inglés TANB) (Bielza y Larrañaga, 2014).

Posteriormente, Codecasa y Stella (2014b) introdujeron más clasificadores pertenecientes a esta clase: el clasificador Max-k clasificador bayesiano en tiempo continuo (con su acrónimo en inglés Max-k CTBNC) y el clasificador Max-k naive Bayes aumentado en tiempo continuo (Max-k ACTNB). El Max-k CTBNC consiste en un CTBNC en el que el número de padres para cada uno de los nodos variable está limitado por $k \in \mathbb{Z}^+$. Y el Max-k ACTNB es un caso particular del Max-k CTBNC en el que el nodo clase ha de pertenecer al conjunto de padres de todos los nodos variable. Este tipo de redes será explicado en mayor profundidad en el siguiente capítulo, por lo que no se entrará en más detalles. Finalmente, se presentan algunas figuras para ilustrar algunos de estos clasificadores (Figura 2.3 y 2.4).

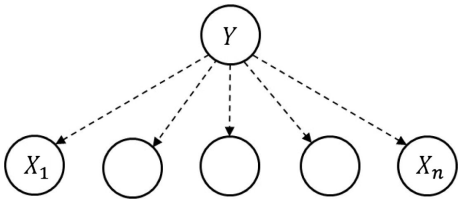

Figura 2.3: Estructura de un CTNB. Nótese que, además, el CTNB es un Max-1 clasificador bayesiano en tiempo continuo, y, concretamente, es un Max-1 naive Bayes aumentado en tiempo continuo. Fuente Stella y Amer (2012).

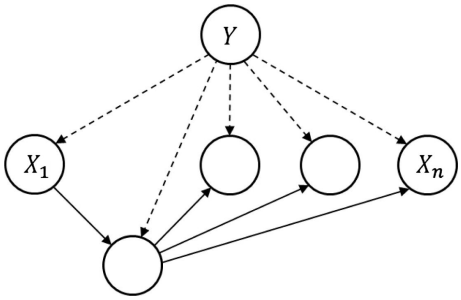

Figura 2.4: Estructura de un CTTANB. Fuente Stella y Amer (2012).

Capítulo 3

Definición del modelo

En este capítulo se va a definir el modelo CTBNCC desde el punto de visto matemático y de la manera más formal posible. Para ello, se va a empezar definiendo el modelo que utiliza como base la cadena de clasificadores, es decir, el CTBNC.

3.1. Clasificadores bayesianos en tiempo continuo

A continuación, se va a definir todos los elementos necesarios para entender el funcionamiento de un CTBNC de acuerdo a lo explicado en Stella y Amer (2012); Codecasa y Stella (2014b).

3.1.1. Redes bayesianas en tiempo continuo

Sea X una v.a. discreta cuyos estados evolucionan de manera continua a lo largo del tiempo, t, y toma valores sobre un dominio, $Val(X) = \{x_1, \ldots, x_m\}$. Podemos representar el proceso homogéneo de Markov, $X(t)$, mediante la siguiente *matriz de intensidad*:

$$\mathbf{Q}_X = \begin{bmatrix} -q_{x_1} & q_{x_1x_2} & \cdots & q_{x_1x_m} \\ q_{x_2x_1} & -q_{x_2} & \cdots & q_{x_2x_m} \\ \cdots & \cdots & \cdots & \cdots \\ q_{x_mx_1} & q_{x_mx_2} & \cdots & -q_{x_m} \end{bmatrix},$$

donde $q_{x_i} = \sum_{j \neq i} q_{x_ix_j}$, para $i = 1, \ldots, m$. Esta matriz describe el comportamiento transitorio de $X(t)$. Si asumimos que $X(0) = x_i$, entonces la v.a. X permanece en el estado x_i una cantidad de tiempo que es una v.a que se distribuye como una exponencial de parámetro q_{x_i}. Por consiguiente, las funciones de densidad y distribución correspondientes a la permanencia de $X(t)$ en el estado x_i son las siguientes, respectivamente:

$$f(t) = q_{x_i} exp(-q_{x_i}t),\ t \geq 0, \quad F(t) = 1 - exp(-q_{x_i}t),\ t \geq 0.$$

De esta manera, el tiempo medio en abandonar el estado x_i es $\dfrac{1}{q_{x_i}}$. Además, sabiendo que va a ocurrir una transición, la probabilidad de pasar al estado x_j es $\dfrac{q_{x_ix_j}}{q_{x_i}}$.

Una CTBN es una red bayesiana (aunque recordemos que permite ciclos) que representa el tiempo de forma explícita. De esta manera, permite recuperar la distribución de probabilidad a lo largo del tiempo cuando ocurre un evento determinado (Codecasa y Stella, 2014b). Para ello, sus nodos se asocian con v.v.a.a. y su estado evoluciona de manera continua a lo largo del tiempo.

Ahora veamos la definición formal introducida por Nodelman *et al.* (2012a). Sea $\mathbf{X}$ un conjunto de v.v.a.a. $\mathbf{X} = \{X_1, \ldots, X_m\}$, donde cada X_n tiene un dominio finito de valores $Val(X_n) = \{x_1, \ldots, x_{I_n}\}$. Una CTBN, $\aleph$, sobre $\mathbf{X}$ consiste en:

- Una distribución inicial $P^0_{\mathbf{X}}$ que se especifica mediante una red bayesiana, $\mathcal{B}$ sobre $\mathbf{X}$.
- Un modelo de transición en tiempo continuo definido mediante:
  - Un grafo dirigido, posiblemente cíclico, cuyos nodos son $X_1, \ldots, X_m$, y donde $\mathbf{Pa}(X_n)$ denota el conjunto de los padres del nodo X_n en el grafo.
  - Una matriz de intensidad condicional, $\mathbf{Q}_{X_n}^{\mathbf{Pa(X_n)}}$, para cada $X_n \in \mathbf{X}$.

Dada la variable aleatoria X_n, la matriz de intensidad condicional (CIM), $\mathbf{Q}_{X_n}^{\mathbf{Pa}(X_n)}$, consiste en un conjunto de matrices de intensidad, $\mathbf{Q}_{X_n}^{\mathbf{pa}(X_n)}$, una para cada instanciación $\mathbf{pa}(X_n)$ de $\mathbf{Pa}(X_n)$:

$$\mathbf{Q}_{X_n}^{\mathbf{pa}(X_n)} = \begin{bmatrix} -q_{x_1}^{\mathbf{pa}(X_n)} & q_{x_1x_2}^{\mathbf{pa}(X_n)} & \ldots & q_{x_1x_{I_n}}^{\mathbf{pa}(X_n)} \\ q_{x_2x_1}^{\mathbf{pa}(X_n)} & -q_{x_2}^{\mathbf{pa}(X_n)} & \ldots & q_{x_2x_{I_n}}^{\mathbf{pa}(X_n)} \\ \ldots & \ldots & \ldots & \ldots \\ q_{x_{I_n}x_1}^{\mathbf{pa}(X_n)} & q_{x_{I_n}x_2}^{\mathbf{pa}(X_n)} & \ldots & -q_{x_{I_n}}^{\mathbf{pa}(X_n)} \end{bmatrix},$$

Se denota $\theta_{x_ix_j}^{\mathbf{pa}(X_n)} = \dfrac{q_{x_ix_j}^{\mathbf{pa}(X_n)}}{q_{x_i}^{\mathbf{pa}(X_n)}}$, que representa la probabilidad de que la variable X_n transicione del estado x_i al x_j, cuando se sabe que la transición ocurre en un instante determinado de tiempo y $\mathbf{Pa}(X_n) = \mathbf{pa}(X_n)$.

Las CTBNs permiten dos tipos distintos de evidencia:

- *Evidencia puntual*: en un instante, t, para un subconjunto de variables $X_1, \ldots, X_k \in \mathbf{X}$, es el conocimiento de los estados de las variables de este subconjunto en ese instante determinado de tiempo. Es decir, $X_1^t = x_1^t, \ldots, X_k^t = x_k^t$, o de manera abreviada: $\mathbf{X}_k^t = \mathbf{x}_k^t$.
- *Evidencia continua*: es el conocimiento de los estados de las variables de un subconjunto de variables, $X_1, \ldots, X_k \in \mathbf{X}$, a lo largo de un intervalo de tiempo, $[t_1, t_2)$. Y se considera que el estado de las variables no cambia a lo largo de ese intervalo. Es decir, $X_1^{[t_1,t_2)} = x_1^{[t_1,t_2)}, \ldots, X_k^{[t_1,t_2)} = x_k^{[t_1,t_2)}$, o de manera abreviada $\mathbf{X}_k^{[t_1,t_2)} = \mathbf{x}_k^{[t_1,t_2)}$.

Aprendizaje de un CTBN

Aprendizaje de la estructura El problema de aprender la estructura de un CTBN fue abordado en (Nodelman *et al.*, 2012c), como el problema de, dado un conjun-

to de datos $\mathcal{D}$, encontrar la estructura, $\mathcal{G}^*$ que maximiza la puntuación bayesiana (Nodelman *et al.*, 2012b):

$$score(\mathcal{G} : \mathcal{D}) = lnP(\mathcal{D}|\mathcal{G}) + lnP(\mathcal{G}). \tag{3.1}$$

No obstante, el espacio de búsqueda del problema de optimización es más simple que el presentan las redes bayesianas y las redes bayesianas dinámicas, ya que todos los arcos son a través del tiempo. Es decir, representan la influencia del valor actual de una variable en el siguiente valor de las otras variables (temporalmente hablando). Por ello, no hay restricciones de aciclicidad y, por consiguiente, es posible optimizar el conjunto de padres de cada variable, $\mathbf{Pa}(X_n)$, de manera independiente. Para un número de padres máximo fijo para cada nodo, la complejidad de aprender la estructura óptima es polinomial respecto al número de variables y la dimensión del conjunto de datos (Codecasa y Stella, 2014b).

Aprendizaje de los parámetros Dado un conjunto de datos $\mathcal{D}$ y una estructura de la CTBN, el aprendizaje de los parámetros se basa en la *estimación marginal de la log-verosimilitud*, y tiene en cuenta las cuentas imaginarias de los hiperparámetros $\alpha_x^{\mathbf{pa}(X)}, \alpha_{xx\prime}^{\mathbf{pa}(X)}$ y $\tau_x^{\mathbf{pa}(X)}$. Los parámetros $q_x^{\mathbf{pa}(X)}$ y $\theta_{xx\prime}^{\mathbf{pa}(X)}$ se estiman de la siguiente manera:

- $\hat{q}_x^{\mathbf{pa}(X)} = \dfrac{\alpha_x^{\mathbf{pa}(X)} + M[x|\mathbf{pa}(X)]}{\tau_x^{\mathbf{pa}(X)} + T[x|\mathbf{pa}(X)]}$
- $\hat{\theta}_{xx\prime}^{\mathbf{pa}(X)} = \dfrac{\alpha_{xx\prime}^{\mathbf{pa}(X)} + M[x, x\prime|\mathbf{pa}(X)]}{\alpha_x^{\mathbf{pa}(X)} + M[x|\mathbf{pa}(X)]}$

donde

- $M[x, x\prime|\mathbf{pa}(X)]$ es el número de veces que X transiciona del estado x al $x\prime$ estando sus padres instanciados a $\mathbf{pa}(X)$. Su cuenta imaginaria es $\alpha_x^{\mathbf{pa}(X)}$.
- $M[x|\mathbf{pa}(X)] = \sum_{x \neq x\prime} M[x, x\prime|\mathbf{pa}(X)]$ es el número de veces que X abandona el estado x cuando sus padres están instanciados a $\mathbf{pa}(X)$. Su cuenta imaginaria es $\alpha_{xx\prime}^{\mathbf{pa}(X)}$.
- $T[x|\mathbf{pa}(X)]$ es la cantidad de tiempo que X pasa en el estado x cuando sus padres están instanciados a $\mathbf{pa}(X)$. Su cuenta imaginaria es $\tau_x^{\mathbf{pa}(X)}$.

Todo ello calculado sobre el conjunto de datos $\mathcal{D}$.

3.1.2. Clasificador bayesiano en tiempo continuo

Un CTBNC es una clase de modelos de clasificación supervisada. Concretamente, es un par $\mathcal{C} = \{\aleph, P(Y)\}$ donde $\aleph$ es un modelo CTBN con nodos variable $X_1, \ldots, X_N$, y un nodo clase Y con probabilidad marginal $P(Y)$ sobre estados $Val(Y) = \{y_1, \ldots, y_K\}$, $\mathcal{G}$ es el grafo del CTBNC (Stella y Amer, 2012), tal que se cumplen las siguientes condiciones:

- $\mathcal{G}$ está conectado.
- $Pa(Y) = \emptyset$, es decir, la variable clase Y está asociada con un nodo raíz.

- Y está completamente determinada por $P(Y)$ y no depende del tiempo.

La Figura 3.1 muestra un ejemplo de CTBNC.

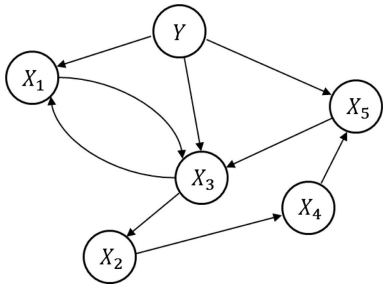

Figura 3.1: Ejemplo de estructura de un CTBNC con 5 nodos variable, $X_1, \ldots, X_5$, y un nodo clase, Y. Fuente Stella y Amer (2012).

Ahora se va a pasar a ver algunos de los modelos pertenecientes a esta clase.

Clasificador naive Bayes en tiempo continuo

Un clasificador naive Bayes en tiempo continuo es un CTBNC $\mathcal{C} = \{\aleph, P(Y)\}$ tal que $\mathbf{Pa}(X_n) = \{Y\}$, para $n = 1, \ldots, N$ (Stella y Amer, 2012).

Clasificador bayesiano aumentado en árbol en tiempo continuo

Un clasificador bayesiano aumentado en árbol en tiempo continuo es un CTBNC $\mathcal{C} = \{\aleph, P(Y)\}$ tal que se cumplen las siguientes condiciones (Stella y Amer, 2012):

- $Y \in Pa(X_n)$, $n = 1, \ldots, N$.
- Los nodos de variables X_n, $n = 1, \ldots, N$ forman un árbol. Es decir, existe un único j tal que $|\mathbf{Pa}(X_j)| = 1$, y para $i \neq j$, $|\mathbf{Pa}(X_i)| = 2$.

Max-k clasificador bayesiano en tiempo continuo

Un Max-k clasificador bayesiano en tiempo continuo es un CTBNC tal que el número de padres de cada variable atributo está acotado por $k \in \mathbb{Z}^+$. Es decir, $|Pa(X_n)| \leq k$, para $n = 1, \ldots, N$ (Codecasa y Stella, 2014b).

Max-k naive Bayes aumentado en tiempo continuo

Un Max-k naive Bayes aumentado en tiempo continuo es un Max-k clasificador bayesiano en tiempo continuo tal que el nodo clase, Y, pertenece al conjunto de padres de cada uno de los nodos variable. Es decir, $Y \in Pa(X_n)$ para $n = 1, \ldots, N$ (Codecasa y Stella, 2014b).

3.1.3. Aprendizaje de un CTBNC

El aprendizaje de un CTBNC a partir de un conjunto de datos consiste en aprender una CTBN en la cual un nodo específico, el nodo clase, Y, no depende del tiempo. Además, ya que el nodo raíz, Y, no tiene padres, el algoritmo se simplifica (Codecasa

y Stella, 2014b). En este caso, se va explicar el aprendizaje siguiendo a Codecasa y Stella (2014a).

Pero antes, para entender el algoritmo que se muestra en la Figura 3.2, es necesario introducir algunos conceptos.

Un *J-time-stream* sobre el intervalo $[t_1, T)$ es una partición del intervalo en J intervalos: $[t_1, t_2), [t_2, t_3), \ldots, [t_J, t_{J+1})$, con $t_{J+1} = T$.

Dada ahora una CTBN con N nodos, y un *J-time-stream*, un *J-evidence-stream* es el conjunto de instanciaciones conjuntas $\mathbf{X} = \mathbf{x}$ para cualquier subconjunto de v.v.a.a., $S = \{X_1, \ldots, X_s\}$, con $1 \leq s \leq n$ asociadas a cada uno de los J intervalos temporales. Un *J-evidence-stream* se denomina *completamente observado* cuando el estado de todas las variables, X_n, se conoce a lo largo de todo el intervalo $[0, T)$. En el caso de no ser completamente observado, se denomina parcialmente observado.

Aprendizaje de la estructura

El problema de aprender un CTBNC se reduce a maximizar la Ecuación (3.1) sujeto a las restricciones sobre la variable clase que introduce el CTBNC (Sección 3.1.2).

Aprendizaje de los parámetros

La única diferencia entre el aprendizaje de los parámetros de la CTBN y el CTBNC es la necesidad de aprender la distribución de probabilidad del nodo clase (Codecasa y Stella, 2014a). Y, al ser un nodo estático, esto puede hacerse de la siguiente manera:

$$\theta_y = \frac{\alpha_y + M[y]}{\sum_{y\prime} \alpha_{y\prime} + M[y\prime]},$$

donde $M[y]$ es el número de trayectorias en las que la variable clase, Y toma el valor y, y α_y son las cuentas imaginarias relacionadas con la variable clase.

La Figura 3.2 presenta el algoritmo de aprendizaje de parámetros de un CTBNC.

Require: a data set $\mathcal{D}$ of *fully observed J-evidence-streams* $\{(\mathbf{x}^1, \mathbf{x}^2, ..., \mathbf{x}^{J_1}); ...; (\mathbf{x}^1, \mathbf{x}^2, ..., \mathbf{x}^{J_{|\mathcal{D}|}})\}$, and the corresponding set of classes $\{y_1, y_2, ..., y_{|\mathcal{D}|}\}$, $y_j \in Val(Y)$, $j = 1, 2, ..., |\mathcal{D}|$.

Ensure: the CTNB classifier $\mathcal{C} = \{\aleph, P(Y)\}$.

1: **for** k=1 to K **do**
2: $\mathrm{P}(k) := 0$;
3: **end for**
4: **for** s=1 to $|\mathcal{D}|$ **do**
5: $\mathrm{P}(y_s) := \mathrm{P}(y_s) + \frac{1}{|\mathcal{D}|}$;
6: **end for**
7: **for** s=1 to $|\mathcal{D}|$ **do**
8: i=1;
9: **while** $t_i \leq T_s$ **do**
10: **for** $n = 1$ to N **do**
11: $M(x_n^i, x_n^{i+1}, y_s) := M(x_n^i, x_n^{i+1}, y_s) + 1$;
12: $T(x_n^i, y_s) := T(x_n^i, y_s) + (t_i - t_{i-1})$;
13: **end for**
14: i:=i+1;
15: **end while**
16: **end for**
17: **for** $k = 1$ to K **do**
18: **for** $n = 1$ to N **do**
19: **for** $x \in Val(X_n)$ **do**
20: $MM(x, y_k) := \sum_{x' \neq x} M(x, x', y_k)$;
21: $q(x, y_k) := \frac{MM(x, y_k)}{T(x, y_k)}$;
22: $\theta(x, y_k) := \frac{M(x, x', y_k)}{MM(x, y_k)}$;
23: **end for**
24: **end for**
25: **end for**
26: **return** $\aleph$, P.

Figura 3.2: Aprendizaje de los parámetros de un CTBN. Las líneas $1 - 6$ calculan la probabilidad a priori de la variable clase, Y (devuelta en la variable P). Las líneas $7-16$ realizan los cálculos estadísticos suficientes para cada variable X_n, con $n = 1, \ldots, N$, sobre el conjunto de datos $\mathcal{D}$, i.e. $M(x, x\prime, y_k)$ y $T(x, y_k)$ que no son otra cosa que los hiperparámetros vistos en la Sección 3.1.1 pero para cada valor de y_k de la variable clase Y. Las líneas $17 - 25$ calculan el correspondiente MLE de los parámetros, que son devueltos en el CTNB, $\aleph$. Fuente Stella y Amer (2012).

3.1.4. Clasificación

Un CTBNC (Codecasa y Stella, 2014b) clasifica un *J-evidence-stream* completamente observado $\mathbf{X}^{[t_1,t_2)} = \mathbf{x}^{[t_1,t_2)} = \mathbf{x}^1, \mathbf{X}^{[t_2,t_3)} = \mathbf{x}^{[t_2,t_3)} = \mathbf{x}^2, \ldots, \mathbf{X}^{[t_J,t_{J+1})} = \mathbf{x}^{[t_J,t_{J+1})} = \mathbf{x}^J$, donde todos los nodos variable están observados, eligiendo el valor, y^*, para la variable clase, Y que maximiza la probabilidad a posteriori (MPE)

$$P(Y|\mathbf{x}^1, \ldots, \mathbf{x}^J) \; \alpha \; P(Y) \prod_{j=1}^{J} q_{x_{[j]}^j x_{[j]}^{j+1}}^{\mathbf{pa}(X_{[j]})} \prod_{n=1}^{N} exp(-q_{x_n^j}^{\mathbf{pa}(X_n)} \delta_j),$$

donde

- $X_{[j]}$ es la variable que transiciona en el instante t_{j+1}.
- $q_{x_{[j]}^j x_{[j]}^{j+1}}^{\mathbf{pa}(X_{[j]})}$ es el parámetro asociado a la transición del estado $x_{[j]}^j$, al estado $x_{[j]}^{j+1}$, de la variable $X_{[j]}$ dada la instanciación de sus padres, $\mathbf{pa}(X_{[j]})$, durante los intervalos j-ésimo y $(j + 1)$-ésimo.

- $q_{x_n^j}^{\mathbf{pa}(X_n)}$ es el parámetro asociado al estado x_n^j en el que la variable X_n estaba en el j-ésimo intervalo dado el estado de sus padres, $\mathbf{pa}(X_n)$, en ese mismo intervalo.

- $\delta_j = t_{j+1} - t_j$ es la longitud del intervalo j-ésimo en el *J-evidence-stream*.

La Figura 3.3 representa el algoritmo de inferencia de un CTBNC.

Require: a CTBNC $\mathcal{C} = \{\aleph, P(Y)\}$ consisting of N feature (i.e. attribute) nodes and a class node Y such that $Val(Y) = \{y_1, y_2, \ldots, y_S\}$, a *fully observed evidence stream* $(\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^J)$.

Ensure: the maximum a posteriori classification y^* for the *fully observed J-evidence-stream* $(\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^J)$.

for $s = 1$ to S **do**
  $\log p(y_s) \leftarrow \log P(y_s)$
end for
for $s = 1$ to S **do**
  for $j = 1$ to J **do**
    for $n = 1$ to N **do**
      $\log p(y_s) := \log p(y_s) - q_{x_n^j}^{pa(X_n)}(t_{j+1} - t_j)$
      if $x_n^j \neq x_n^{j+1}$ **then**
        $\log p(y_s) := \log p(y_s) + \log(q_{x_n^j x_n^{j+1}}^{pa(X_n)})$
      end if
    end for
  end for
end for
$y^* \leftarrow \arg\max_{y \in Val(Y)} \log p(y)$.
return y^*

Figura 3.3: Algoritmo de inferencia para un CTBNC. Fuente Codecasa y Stella (2014b).

Para más información acerca del algoritmo de inferencia, ver Stella y Amer (2012); Codecasa y Stella (2014a,b). Concretamente, este algoritmo se explica en mayor detalle Stella y Amer (2012), aunque, en Codecasa y Stella (2014b) se hacen algunos ajustes que terminan derivando en el algoritmo que podemos ver en la Figura 3.3, y que es el que usa el software utilizado en los experimentos (Codecasa y Stella, 2014a).

3.2. Clasificadores en cadena

Dado un problema de clasificación multi-etiqueta, con un conjunto de etiquetas binarias finito, Y, con $|Y| = d$. Read *et al.* (2011) define una cadena de clasificadores (CC) como d clasificadores binarios, donde cada uno de los clasificadores trata de resolver el problema de clasificación binario asociado a la etiqueta $y_j \in Y$ para $j = 1, \ldots, d$. Pero, el espacio de variables predictoras de cada clasificador de la cadena se extiende con las etiquetas predichas por los clasificadores previos en la cadena.

De manera más formal, tomando como muestras de entrenamiento $(\mathbf{x}, S)$, donde $S \subseteq Y \in \{0,1\}^d$, y $\mathbf{x}$ es una instancia del vector de variables predictoras, se define el

entrenamiento de la cadena de clasificadores como sigue (Algoritmo 1):

Algorithm 1: Entrenamiento de acuerdo a Read *et al.* (2011).

Input : $\mathcal{D} = ((\mathbf{x}^1, S^1), ..., (\mathbf{x}^n, S^n))$
for $j \in 1, \ldots, d$ **do**
  /* Transformación y entrenamiento de una etiqueta */
  $\mathcal{D}\prime \leftarrow \{\}$
  for $(\mathbf{x}, S) \in \mathcal{D}$ **do**
    $\mathcal{D}\prime \leftarrow \mathcal{D}\prime \cup ((\mathbf{x}, y_1, \ldots, y_{j-1}), y_j)$
  end
  /* Entrenar el clasificador C_j para predecir la relevancia binaria de y_j */
  $C_j : \mathcal{D}\prime \rightarrow y_j \in \{0, 1\}$
end

Por consiguiente, en el caso $S = Y$, se forma una cadena de clasificadores binarios $C_1, \ldots, C_d$. De esta manera, el proceso de clasificación empieza en C_1 determinando $P(y_1|\mathbf{x})$ y cada uno de los clasificadores siguientes predice $P(y_j|\mathbf{x}, y_1, \ldots, y_{j-1})$ para $j = 2, \ldots, d$, véase el Algoritmo 2.

Algorithm 2: Clasificación de acuerdo a Read *et al.* (2011).

/* Se inicializa el conjunto de etiquetas */
$Y \leftarrow \{\}$
for $j \leftarrow 1$ **to** d **do**
  $Y \leftarrow Y \cup (y_j \leftarrow C_j : (\mathbf{x}, y_1, \ldots, y_{j-1}))$
end
/* La muestra clasificada */
return $(\mathbf{x}, Y)$

Toda esta definición se puede generalizar al problema de clasificación multidimensional simplemente tomando los clasificadores de la cadena como clasificadores multiclase sin mayor dificultad.

De esta manera se supera, tal y como se comentaba en el Capítulo 2, el problema de la independencia entre etiquetas de la relevancia binaria, pero el orden de las etiquetas en la predicción es altamente determinante de cara al resultado.

Finalmente, se presenta una figura de las cadenas de clasificadores bayesianos vistos en el Capítulo 2.

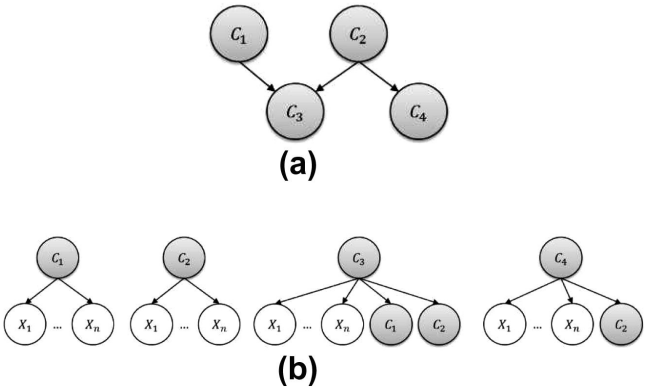

Figura 3.4: Estructura de una cadena de clasificadores bayesianos: (a): red bayesiana que representa las dependencias entre variables clase; (b): conjunto de clasificadores naive Bayes encadenados relativos a la estructura presentada en (a). Fuente Sucar *et al.* (2014).

Desde el punto de vista de las redes bayesianas, y, tal y como veíamos en el Capítulo 2, este tipo de clasificadores en cadena pueden además explotar las ventajas de las redes bayesianas. De esta manera, el cálculo que veíamos en el Algoritmo 2, se simplifica ya que cada variable clase solo depende de las variables clase que son sus padres en la red bayesiana (vista en la Figura 3.4). Y como resultado, se obtiene la fórmula para la clasificación

$$P(\mathbf{Y}|\mathbf{x}) = \prod_{i=1}^{d} P(Y_i|\mathbf{pa}(Y_i), \mathbf{x}).$$

3.3. Cadena de clasificadores bayesianos en tiempo continuo

Una vez definido el clasificador base de la cadena (CTBNC) y lo que es una cadena de clasificadores, el CTBNCC consiste, simplemente, en utilizar una cadena de estos clasificadores.

Capítulo 4

Implementación

En este capítulo se van a repasar los detalles de la implementación del proyecto. Concretamente, se va a ver el sistema operativo utilizado, el lenguaje utilizado, las librerías empleadas y los ficheros que han sido necesarios.

El sistema operativo sobre el que se han implementado los experimentes ha sido:

- Edición: Windows 10 Pro.
- Versión: 2004.
- Versión del sistema operativo: 19041.1052.

Para el desarrollo del proyecto se ha utilizado Anaconda con las siguientes versiones:

- anaconda: 2019.10.
- anaconda-client: 1.7.2.
- anaconda-navigator: 1.9.7.
- anaconda-project: 0.8.3.

En cuanto al lenguaje, se ha utilizado Python 3.7.4. Todo el desarrollo se ha hecho sobre notebooks de Jupyter con las siguientes versiones:

- jupyter_client: 6.1.12.
- jupyter_core: 4.7.1.

A su vez, también se ha utilizado Java para correr la aplicación que se comentará más adelante. Concretamente, se ha utilizado:

- java version "1.8.0_241"
- Java(TM) SE Runtime Environment (build 1.8.0_241-b07)
- Java HotSpot(TM) 64-Bit Server VM (build 25.241-b07, mixed mode)

Todo el proyecto ha sido desarrollado en un entorno virtual con la versión de Python antes mencionada y los paquetes instalados listados en el Anexo 6.

La implementación del clasificador base, CTBNC, que se ha utilizado es el CTBN-CToolkit (Codecasa y Stella, 2014a) disponible en este repositorio de GitHub (Codecasa y Stella, 2014a). Una vez descargada la aplicación, se ha generado un ".jar" para

ejecutarla como una aplicación Java desde los notebook de Jupyter. La cadena de clasificadores se ha implementado utilizando Python para ejecutar la aplicación Java de manera dinámica con los parámetros necesarios en cada caso. Esto es, variando el conjunto de atributos a utilizar en cada clasificador de la cadena, pasando las predicciones de los clasificadores de la cadena a los clasificadores posteriores.

La parte más relevante de la implementación ha sido sin duda entender el funcionamiento de la aplicación de Java debido a las dimensiones de la misma (tiene una enorme cantidad de ficheros que ha sido necesario estudiar de manera exhaustiva para ser capaces de ejecutarla de manera correcta) y el gran número de parámetros que esta utiliza. Una vez, estudiado el funcionamiento de la aplicación y tomando ideas de Codecasa y Stella (2014b), se ha procedido a implementar los experimentos en notebooks de Jupyter tal y como se comentaba anteriormente.

No obstante, aquí ha surgido el problema de la capacidad de cómputo limitada, debida al uso de un ordenador personal que ha llevado a adaptar los experimentos como veremos en el siguiente capítulo para poder mostrar el sentido de las cadenas de clasificadores bayesianos en tiempo continuo.

Específicamente, el problema de la capacidad de cómputo ha llevado a utilizar modelos muy simples que obtenían una precisión muy baja, y por ello se han adaptado los experimentos. Por ejemplo, si se quería ejecutar el software como se menciona en los ejemplos de ejecución proporcionados en Codecasa y Stella (2014a), el modelo para clasificar una única clase tardaba más de 24 horas en ejecutarse, y por lo tanto, ha habido que adaptarlo. Un ejemplo de esto sería ejecutar:

```
java -jar CTBNCToolkit.jar -CTBNC=CTBNC4-LL,penalty <data>
```

Y el simple hecho de añadir "penalty", que no es otra cosa que añadir una penalización a la dimensionalidad a la hora de aprender el modelo, hacía que la ejecución tardara más de 24 horas para una única clase.

Por consiguiente, se han tenido que ejecutar los modelos de la manera más sencilla posible, tardando igualmente del orden de 12 horas la ejecución de la cadena entera, con el fin de poder terminar los experimentos en el tiempo requerido.

Como se comentaba anteriormente, los parámetros para los modelos se han ido generando de manera dinámica, y además, utilizando Python se ha tenido que ir pasando la información de un clasificador de la cadena a los siguientes. Para ello, se volcaban los datos de clasificación en un directorio de resultados, y, posteriormente, utilizando la librería `pandas` se han manejado los resultados de manera que ha sido posible pasárselos a los siguientes clasificadores.

Un ejemplo de ejecución dinámica utilizada sería:

```
!java -jar CTBNCToolkit.jar -CTBNC={mod} -validation=HO,{train_perc}
-validColumns={features_cmd} -trjSeparator={item} -timeName={time}
-className={item} -rPath={outDir} {tmpDir}
```

Todos los parámetros que aparecen entre llaves se generan de manera dinámica para el correcto funcionamiento de la cadena. Una parte también muy pesada de la implementación ha sido la gestión de los resultados y conseguir automatizar el paso de predicciones de cada clasificador al siguiente, así como el cálculo de los resultados de la clasificación. Esto ha sido debido, principalmente, a que la aplicación de Java

no tenía ni de lejos preparación alguna para realizar este tipo de tareas.

Para más información, véase el código disponible en GitHub que tiene, básicamente, la plantilla seguida para los distintos experimentos realizados, además de las referencias a los repositorios que se han estudiado.

Capítulo 5

Experimentos

En este capítulo se van a describir los experimentos realizados y los resultados obtenidos en los mismos. Vale la pena destacar que debido a la baja capacidad de cómputo del ordenador personal (tal y como se ha mencionado en el Capítulo 4) utilizado para llevar a cabo los experimentos, no se han podido ejecutar experimentos más complejos que se propondrán como posibles líneas de trabajo futuras en el Capítulo 6.

También hay que mencionar que se han utilizado CTBNCs con clasificadores muy sencillos de aprender y no se ha complicado el uso de parámetros ya que el alcance del proyecto era mostrar que las cadenas de clasificadores en tiempo continuo tienen sentido, y funcionan tal y como se esperaba.

Esta parte también ha tenido repercusiones en los conjuntos de datos utilizados. En un inicio se planteó, debido a la falta de conjuntos de datos de dominio público para este tipo de problemas, generar conjuntos de datos sintéticos. Por ello, se generaron una serie de conjuntos de datos sintéticos, concretamente cinco de ellos. Cada uno representando los siguientes problemas:

- Conjunto de datos 1: conjunto de datos multi-etiqueta con muchas dependencias entre las variables clase, y muchas dependencias entre las variables predictoras.
- Conjunto de datos 2: conjunto de datos multi-etiqueta con variables clase independientes entre sí, y muchas dependencias entre las variables predictoras.
- Conjunto de datos 3: conjunto de datos multidimensional con muchas dependencias entre las variables clase.
- Conjunto de datos 4: conjunto de datos multi-etiqueta con muchas dependencias entre las variables clase, y muy pocas dependencias entre las variables predictoras.
- Conjunto de datos 5: conjunto de datos multi-etiqueta con variables clase independientes entre sí, y muy pocas dependencias entre las variables predictoras.

Todos estos conjuntos de datos contienen 2000 trayectorias, un conjunto de 10 variables predictoras y 10 variables clase.

Por último, se ha utilizado un conjunto de datos real de carácter confidencial perteneciente al sector de la energía con 3633 trayectorias multidimensionales, con 6 variables clase y 15 variables predictoras en el cual existen bastantes dependencias entre las variables clase. Este conjunto de datos es del estilo del conjunto de datos 3. Por ello, no se darán más detalles sobre el mismo.

En cuanto a los modelos utilizados, estos son básicamente los explicados en el Capítulo 3, y aparecen en las tablas de resultados con la siguiente notación:

- CTNB: clasificador naive Bayes en tiempo continuo.
- CTBNCk: Max-k clasificador bayesiano en tiempo continuo.

Además, se añade un guión seguido de CC en el caso de la cadena de clasificadores, y un guión seguido de BR en el caso de la relevancia binaria.

En cuanto a las métricas utilizadas, dado que los conjuntos de datos están balanceados, se ha utilizado la precisión sobre cada una de las clases, y la precisión sobre todas las clases a la vez, i.e. el porcentaje de muestras para las que se predicen todas las etiquetas correctamente. Nótese que esta última depende mucho del número de variables clase del conjunto de datos utilizado.

En todos los experimentos se han utilizado el $75\,\%$ de las trayectorias para entrenar el modelo, y el $25\,\%$ restante para comprobar su funcionamiento. Los resultados de precisión que se muestran en las tablas de resultados son, por consiguiente, sobre ese $25\,\%$ de trayectorias. El resto de parámetros utilizados son los predeterminados del CTBNCToolkit. Para más información ver Codecasa y Stella (2014a). También se realizó validación cruzada, disponible en el CTBNCToolkit, pero debido a la gran cantidad de tiempo que tardaba la ejecución de los experimentos, se acabaron utilizando los conjunto de *train* y *test* tradicionales con los porcentajes anteriormente mencionados.

También, vale la pena mencionar que los experimentos han sido realizados para el Max-k naive Bayes aumentado en tiempo continuo, pero no se incluyen los resultados en las tablas porque han sido exactamente los mismos que para el CTNB. Esto se debe a que se ha tomado $k = 2$, dada la capacidad de cómputo disponible.

5.1. Experimento 1: Experimento de referencia

5.1.1. Descripción del experimento

Para el primer experimento se quería mostrar que los CTBNCC tenían sentido en el caso del tiempo continuo. Para ello, se decidió comparar los resultados con los obtenidos en la relevancia binaria. No obstante, y como se muestra en los resultados iniciales, surgió el problema de que debido a la baja precisión de los clasificadores sobre clases individuales, el resultado sobre todos los conjuntos de datos utilizados era prácticamente el mismo para la cadena de clasificadores que para la relevancia binaria. Este problema se producía tanto para el conjunto de datos real como para todos los conjuntos de datos sintéticos. No obstante, se pueden ver las diferencias en la precisión global para los conjuntos de datos sintéticos.

5.1.2. Resultados del experimento

Por simplicidad solo se incluyen los resultados obtenidos para el conjunto de datos real. Los resultados obtenidos se muestran en los Cuadros 5.1, 5.2, 5.3, 5.4, 5.5 y 5.6.

Modelo	**Precisión sobre clases**						**Precisión global**
	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	
CTNB-CC	0.58	0.58	0.62	0.71	0.75	0.75	0.36
CTNB-BR	0.58	0.58	0.62	0.71	0.75	0.75	0.36
CTBNC2-CC	0.58	0.58	0.62	0.71	0.75	0.75	0.36
CTBNC2-BR	0.58	0.58	0.62	0.71	0.75	0.75	0.36

Cuadro 5.1: Resultados del primer experimento para el conjunto de datos real. BR = relevancia binaria; CC = cadena de clasificadores.

Modelo	**Precisión sobre clases**										**Precisión global**
	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7	Y_8	Y_9	Y_{10}	
CTNB-CC	0.86	0.84	0.72	0.71	0.60	0.55	0.50	0.51	0.52	0.53	0.012
CTNB-BR	0.86	0.84	0.72	0.71	0.60	0.55	0.50	0.51	0.52	0.53	0.012
CTBNC2-CC	0.89	0.82	0.88	0.79	0.55	0.58	0.61	0.50	0.64	0.53	0.016
CTBNC2-BR	0.89	0.88	0.69	0.71	0.60	0.58	0.61	0.50	0.64	0.53	0.012

Cuadro 5.2: Resultados del primer experimento para el conjunto de datos sintético 1. BR = relevancia binaria; CC = cadena de clasificadores.

Modelo	**Precisión sobre clases**										**Precisión global**
	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7	Y_8	Y_9	Y_{10}	
CTNB-CC	0.80	0.71	0.65	0.50	0.65	0.56	0.52	0.50	0.49	0.52	0.002
CTNB-BR	0.80	0.71	0.65	0.50	0.65	0.56	0.52	0.50	0.49	0.52	0.002
CTBNC2-CC	0.83	0.82	0.73	0.56	0.90	0.81	0.67	0.68	0.55	0.64	0.024
CTBNC2-BR	0.89	0.88	0.69	0.71	0.60	0.58	0.61	0.50	0.64	0.53	0.024

Cuadro 5.3: Resultados del primer experimento para el conjunto de datos sintético 2. BR = relevancia binaria; CC = cadena de clasificadores.

Modelo	**Precisión sobre clases**										**Precisión global**
	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7	Y_8	Y_9	Y_{10}	
CTNB-CC	0.72	0.47	0.43	0.36	0.34	0.49	0.52	0.51	0.49	0.50	0.004
CTNB-BR	0.72	0.47	0.42	0.36	0.35	0.49	0.52	0.51	0.49	0.50	0.004
CTBNC2-CC	0.76	0.64	0.54	0.42	0.36	0.52	0.50	0.53	0.50	0.53	0.004
CTBNC2-BR	0.76	0.68	0.54	0.44	0.36	0.52	0.50	0.53	0.50	0.53	0.002

Cuadro 5.4: Resultados del primer experimento para el conjunto de datos sintético 3. BR = relevancia binaria; CC = cadena de clasificadores.

Modelo	Precisión sobre clases										Precisión global
	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7	Y_8	Y_9	Y_{10}	
CTNB-CC	0.87	0.67	0.73	0.64	0.61	0.63	0.75	0.77	0.59	0.61	0.022
CTNB-BR	0.87	0.67	0.73	0.63	0.61	0.63	0.75	0.77	0.60	0.61	0.022
CTBNC2-CC	0.89	0.80	0.84	0.82	0.75	0.74	0.89	0.92	0.80	0.85	0.123
CTBNC2-BR	0.89	0.78	0.84	0.82	0.75	0.74	0.89	0.92	0.70	0.85	0.012

Cuadro 5.5: Resultados del primer experimento para el conjunto de datos sintético 4. BR = relevancia binaria; CC = cadena de clasificadores.

Modelo	Precisión sobre clases										Precisión global
	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7	Y_8	Y_9	Y_{10}	
CTNB-CC	0.81	0.69	0.66	0.74	0.69	0.65	0.74	0.77	0.65	0.74	0.022
CTNB-BR	0.81	0.69	0.66	0.74	0.69	0.65	0.74	0.77	0.64	0.74	0.022
CTBNC2-CC	0.84	0.82	0.83	0.91	0.84	0.79	0.86	0.87	0.76	0.87	0.19
CTBNC2-BR	0.84	0.82	0.83	0.91	0.84	0.79	0.86	0.87	0.76	0.87	0.19

Cuadro 5.6: Resultados del primer experimento para el conjunto de datos sintético 5. BR = relevancia binaria; CC = cadena de clasificadores.

Podemos observar que como los clasificadores sobre clases individuales no son lo suficientemente buenos, los resultados en la relevancia binaria son prácticamente los mismos que los resultados en la cadena de clasificadores en la predicción de clases individuales. No obstante, la precisión global muestra que las cadenas de clasificadores tienen sentido. Ya que en los conjuntos de datos sintéticos con dependencias entre variables clase, i.e. los conjuntos de datos 1, 3 y 4, la cadena de clasificadores ya muestra mejor precisión global que la relevancia binaria.

5.2. Experimento 2: Clasificadores perfectos

5.2.1. Descripción del experimento

Para el segundo experimento, y a la luz de los resultados anteriores, se decidió implementar la cadena de clasificadores, pero suponiendo que las clases predichas que se pasan a los clasificadores posteriores en la cadena son perfectas. Es decir, suponiendo que dado el j-ésimo clasificador de la cadena, que recordemos que utiliza como conjunto de variables predictoras extendido todos las variables predictoras iniciales más las predicciones de los clasificadores anteriores, estas $j-1$ predicciones (de los clasificadores anteriores) son correctas, i.e. el valor predicho es el real. Esto con el objetivo de ver cuánto podía mejorar la cadena de clasificadores las precisiones de la relevancia binaria cuando los clasificadores individuales tienen una precisión óptima. Aunque además de depender de la precisión, como es lógico dada la naturaleza del modelo, esta mejora también depende de las relaciones entre variables clase.

A partir de este experimento, dada la capacidad de cómputo limitada y el tiempo disponible, los experimentos se han realizado sobre el conjunto de datos real.

5.2.2. Resultados del experimento

Con esta asunción, los resultados obtenidos se muestran en el Cuadro 5.2.2.

Modelo	Precisión sobre clases						Precisión global
	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	
CTNB-CC	0.58	0.998	0.75	0.80	0.83	0.994	0.41
CTNB-BR	0.58	0.58	0.62	0.71	0.75	0.75	0.36
CTBNC2-CC	0.58	0.998	0.62	0.80	0.75	0.75	0.37
CTBNC2-BR	0.58	0.58	0.62	0.71	0.75	0.75	0.36

Cuadro 5.7: Resultados del segundo experimento. BR = relevancia binaria; CC = cadena de clasificadores.

Esta tabla ya nos muestra que los clasificadores en cadena tienen sentido en tiempo continuo ya que mejoran la precisión de manera notable teniendo en cuenta las dependencias entre variables clase.

5.3. Experimento 3: Orden de las variables en la cadena

5.3.1. Descripción del experimento

Finalmente, para el tercer experimento, se decidió comprobar otra de las propiedades de las cadenas de clasificadores en el caso estático. Para ello, se procedió a realizar de nuevo los experimentos anteriores, pero en este caso alterando el orden de las variables de los clasificadores. Concretamente, se quería mostrar que al alterar el orden de las variables en los clasificadores los resultados de precisión sobre clases individuales se verían afectados debido a la posible interpretación de otras relaciones entre variables clase. Se podrían haber implementado órdenes aleatorios y otras variaciones como las vistas en el Capítulo 2, pero dada la baja capacidad de cómputo y el alcance del proyecto, simplemente se ha tomado el orden inverso de las variables clase para comprobar si variaba la precisión.

5.3.2. Resultados del experimento

Los resultados obtenidos se muestran en el Cuadro 5.3.2:

Modelo	Precisión sobre clases						Precisión global
	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	
CTNB-CC	0.998	0.74	0.74	0.86	0.95	0.75	0.43
CTNB-BR	0.58	0.58	0.62	0.71	0.75	0.75	0.36
CTBNC2-CC	0.998	0.58	0.62	0.71	0.75	0.75	0.36
CTBNC2-BR	0.58	0.58	0.62	0.71	0.75	0.75	0.36

Cuadro 5.8: Resultados del tercer experimento. BR = relevancia binaria; CC = cadena de clasificadores.

Estos resultados nos muestran que, efectivamente, el orden de los clasificadores de la cadena es totalmente relevante en el caso de tiempo continuo, tal y como lo era en el caso estático.

5.4. Conclusiones

En general, los resultados muestran que aunque la distribución de las variables clase no cambia en el tiempo, el conocer su valor cuando estas están relacionadas con el resto de variables clase, es decir, al utilizarlas como variables predictoras para los siguientes clasificadores de la cadena, permite mejorar las predicciones con respecto a la relevancia binaria, tal y como pasa con las cadenas de clasificadores estáticos.

Además, y como es lógico, es importante que los clasificadores obtengan buenos resultados al predecir sus clases para que los clasificadores posteriores de la cadena también mejoren sus resultados, tal y como pasaba en el caso estático.

Por otro lado, los experimentos también muestran, que salvo en casos muy particulares en los que un clasificador de la cadena tiene una precisión muy baja, la cadena de clasificadores es al menos tan buena como la relevancia binaria. Lo cual vuelve a coincidir con lo que se veía en el Capítulo 2 para el caso estático.

Por último, el tercer experimento muestra que, de nuevo, tal y como pasaba en el caso estático, el orden de los clasificadores de la cadena es totalmente relevante para el resultado de la clasificación.

Así que, básicamente, se puede concluir que las cadenas de clasificadores en tiempo continuo que utilizan un CTBNC como clasificador base, tienen el mismo sentido que las cadenas de clasificadores en el caso estático.

Capítulo 6

Conclusiones y trabajo futuro

En este capítulo se van a exponer las conclusiones obtenidas de la realización del trabajo, así como algunas posibles líneas de trabajo futuro.

Como conclusiones del trabajo lo primero es comentar que la parte de mayor carga del mismo ha sido el estudio de gran cantidad de artículos. Esto ha sido principalmente debido a que las cadenas de clasificadores en tiempo continuo no existen en la literatura actual, en cuanto a clasificadores bayesianos se refiere (lo cual suponía un reto de diseño de un nuevo modelo).

Otra parte que ha supuesto una gran carga de trabajo ha sido el estudio del funcionamiento de la aplicación de Java (Codecasa y Stella, 2014a), tal y como se comentaba en el Capítulo 4, además de su integración utilizando código en Python para funcionar como una cadena de clasificadores, y ser capaz de obtener los resultados de clasificación.

Un gran contratiempo en el desarrollo del proyecto ha sido, sin duda alguna, la ausencia de datos, de dominio público, de problemas de clasificación multidimensionales de series temporales multivariadas.

Pero lo más relevante de este trabajo ha sido que finalmente, a pesar de los contratiempos de la capacidad de cómputo limitada de la que se disponía y la ausencia de conjuntos de datos para este problema de clasificación, se ha conseguido mostrar que las cadenas de clasificadores bayesianos en tiempo continuo tienen sentido y, de hecho, cumplen con el funcionamiento esperado que se ha comprobado en los experimentos de las cadenas de clasificadores en el caso estático.

No obstante, han quedado muchas líneas de trabajo futuro abiertas. A continuación se proponen algunas de ellas:

- Realización de un mayor número de experimentos sobre otros conjuntos de datos reales. En este trabajo no ha sido posible debido a que en la actualidad no se encuentran prácticamente conjuntos de datos para clasificación multidimensional de series temporales multivariadas. O al menos no son de dominio público. De hecho, el conjunto de datos real utilizado en este trabajo es de carácter confidencial tal y como se comentaba anteriormente.
- Probar otras implementaciones del clasificador base, CTBNC. O, incluso, desarrollar una nueva implementación en un lenguaje más utilizado por la comuni-

dad para este tipo de tareas, preferiblemente en Python.

- Utilizar un equipo con suficiente capacidad de cómputo para realizar experimentos con clasificadores más complejos que permitan variar los parámetros de los modelos de Codecasa y Stella (2014a) e incremetar el valor de k en los clasificadores Max-k.
- Probar variaciones de las cadenas de clasificadores como las cadenas de clasificadores circulares (Rivas *et al.*, 2018), o las cadenas de clasificadores dinámicas (Kulessa y Mencía, 2018) vistas en el Capítulo 2.
- Probar alternativas más pesadas computacionalmente como los "ensembles" de clasificadores Read *et al.* (2011).
- Aplicar el trabajo realizado en Sucar *et al.* (2014) al caso de clasificadores bayesianos continuos, para comprobar que todas las conclusiones que se extraían para cadenas de clasificadores bayesianos en el caso estático se mantienen.

Bibliografía

Bielza, C., Li, G. y Larrañaga, P. (2011). Multi-dimensional classification with Bayesian networks. *International Journal of Approximate Reasoning*, 52(6), 705-727.

Bielza, C. y Larrañaga, P. (2014). Discrete Bayesian network classifiers: A survey. *ACM Computing Surveys*, 47(1), 1-43.

Boutell, M. R., Luo, J., Shen, X. y Brown, C. M. (2004). Learning multi-label scene classification. *Pattern Recognition*, 37(9), 1757-1771.

Codecasa, D. y Stella, F. (2014). CTBNCToolkit: Continuous time Bayesian network classifier toolkit. *arXiv preprint arXiv:1404.4893*.

Codecasa, D. y Stella, F. (2014). Learning continuous time Bayesian network classifiers. *International Journal of Approximate Reasoning*, 55(8), 1728-1746.

Dagum, P., Galper, A. y Horvitz, E. (1992). Dynamic network models for forecasting. In *Uncertainty in Artificial Intelligence* (pp. 41-48). Morgan Kaufmann.

Dean, T. y Kanazawa, K. (1989). A model for reasoning about persistence and causation. *Computational Intelligence*, 5(2), 142-150.

Gil-Begue, S., Bielza, C. y Larrañaga, P. (2021). Multi-dimensional Bayesian network classifiers: A survey. *Artificial Intelligence Review*, 54(1), 519-559.

Gunawardana, A., Meek, C. y Xu, P. (2011). A model for temporal dependencies in event streams. In *Advances in Neural Information Processing Systems*, 24, 1962-1970.

Kulessa, M. y Mencía, E. L. (2018). Dynamic classifier chain with random decision trees. In *International Conference on Discovery Science* (pp. 33-50). Springer.

Mencía, L. (2020). Extreme gradient boosted multi-label trees for dynamic classifier chains. *arXiv preprint arXiv:2006.08094*.

Nodelman, U., Shelton, C. R. y Koller, D. (2002). Continuous time bayesian networks. In *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence* (pp. 378-387).

Nodelman, U., Shelton, C. R. y Koller, D. (2002). Learning continuous time bayesian networks. In *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence* (pp. 451-458).

Nodelman, U., Shelton, C. R. y Koller, D. (2012). Expectation maximization and complex duration distributions for continuous time Bayesian networks. *arXiv preprint arXiv:1207.1402*.

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2), 257-286.

Rajaram, S., Graepel, T. y Herbrich, R. (2005). Poisson-networks: A model for structured Poisson processes. In *International Workshop on Artificial Intelligence and Statistics* (pp. 277-284).

Read, J., Pfahringer, B., Holmes, G. y Frank, E. (2011). Classifier chains for multi-label classification. *Machine Learning*, 85(3), 333.

Rivas, J. J., Orihuela-Espina, F. y Succar, L. E. (2018). Circular chain classifiers. In *International Conference on Probabilistic Graphical Models* (pp. 380-391).

Simma, A. (2010). *Modeling Events in Time Using Cascades of Poisson Processes* (Doctoral dissertation, UC Berkeley).

Simma, A., Goldszmidt, M., MacCormick, J., Barham, P., Black, R., Isaacs, R. y Mortier, R. (2012). CT-NOR: Representing and reasoning about events in continuous time. *arXiv preprint arXiv:1206.3280*.

Stella, F. y Amer, Y. (2012). Continuous time Bayesian network classifiers. *Journal of Biomedical Informatics*, 45(6), 1108-1119.

Sucar, L. E., Bielza, C., Morales, E. F., Hernandez-Leal, P., Zaragoza, J. H. y Larrañaga, P. (2014). Multi-label classification with Bayesian network-based chain classifiers. *Pattern Recognition Letters*, 41, 14-22.

Tucker, A., t Hoen, P. A., Vinciotti, V. y Liu, X. (2006). Temporal Bayesian classifiers for modelling muscular dystrophy expression data. *Intelligent Data Analysis*, 10(5), 441-455.

van der Gaag, L. C. y de Waal, P. R. (2006). Multi-dimensional Bayesian network classifiers. In *Proceedings of the 3rd European Workshop in Probabilistic Graphical Models*, pp 107–114.

Weiss, J. C. y Page, D. (2013). Forest-based point process for event prediction from electronic health records. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases* (pp. 547-562). Springer.

Zaragoza, J. C., Sucar, E., Morales, E., Bielza, C. y Larrañaga, P. (2011). Bayesian chain classifiers for multidimensional classification. In *Twenty-Second International Joint Conference on Artificial Intelligence*.

Anexo

Este anexo presenta un listado con los paquetes instalados en el entorno virtual y sus versiones.

- argon2-cffi @ file:///C:/ci/argon2-cffi_1613038019788/work.
- async-generator==1.10.
- attrs @ file:///tmp/build/80754af9/attrs_1604765588209/work.
- backcall @ file:///home/ktietz/src/ci/backcall_1611930011877/work.
- bleach @ file:///tmp/build/80754af9/bleach_1612211392645/work.
- certifi==2020.12.5.
- cffi @ file:///C:/ci/cffi_1613247308275/work.
- colorama @ file:///tmp/build/80754af9/colorama_1607707115595/work.
- decorator @ file:///tmp/build/80754af9/decorator_1617624478095/work.
- defusedxml @ file:///tmp/build/80754af9/defusedxml_1615228127516/work.
- entrypoints==0.3.
- importlib-metadata @ file:///C:/ci/importlib-metadata_1615900693889/work.
- ipykernel @ file:///C:/ci/ipykernel_1596208728219/work/dist/ipykernel-5.3.4-py3-none-any.whl.
- ipython @ file:///C:/ci/ipython_1617121109687/work.
- ipython-genutils @ file:///tmp/build/80754af9/ipython_genutils_1606773439826/work.
- jedi==0.17.0.
- Jinja2 @ file:///tmp/build/80754af9/jinja2_1612213139570/work.
- joblib==1.0.1.
- jsonschema @ file:///tmp/build/80754af9/jsonschema_1602607155483/work.
- jupyter-client @ file:///tmp/build/80754af9/jupyter_client_1616770841739/work.
- jupyter-core @ file:///C:/ci/jupyter_core_1612213516947/work.
- jupyterlab-pygments @ file:///tmp/build/80754af9/jupyterlab_pygments_16014-90720602/work.
- llvmlite==0.36.0.

- MarkupSafe @ file:///C:/ci/markupsafe_1594405949945/work.
- mistune @ file:///C:/ci/mistune_1594373272338/work.
- nbclient @ file:///tmp/build/80754af9/nbclient_1614364831625/work.
- nbconvert @ file:///C:/ci/nbconvert_1601914921407/work.
- nbformat @ file:///tmp/build/80754af9/nbformat_1617383369282/work.
- nest-asyncio @ file:///tmp/build/80754af9/nest-asyncio_1613680548246/work.
- notebook @ file:///C:/ci/notebook_1611348264852/work.
- numba==0.53.1.
- numpy==1.20.2.
- packaging @ file:///tmp/build/80754af9/packaging_1611952188834/work.
- pandas==1.2.3.
- pandocfilters @ file:///C:/ci/pandocfilters_1605102427207/work.
- parso @ file:///tmp/build/80754af9/parso_1617223946239/work.
- patsy==0.5.1.
- pickleshare @ file:///tmp/build/80754af9/pickleshare_1606932040724/work.
- prometheus-client @ file:///tmp/build/80754af9/prometheus_client_161738-3679942/work.
- prompt-toolkit @ file:///tmp/build/80754af9/prompt-toolkit_1616415428029/work.
- pycparser @ file:///tmp/build/80754af9/pycparser_1594388511720/work.
- Pygments @ file:///tmp/build/80754af9/pygments_1615143339740/work.
- pyparsing @ file:///home/linux1/recipes/ci/pyparsing_1610983426697/work.
- pyrsistent @ file:///C:/ci/pyrsistent_1600123688363/work.
- python-dateutil @ file:///home/ktietz/src/ci/python-dateutil_1611928101742/work.
- pytz==2021.1.
- pywin32==227.
- pywinpty==0.5.7.
- pyzmq==20.0.0.
- scikit-learn==0.24.1.
- scipy==1.6.2.
- Send2Trash @ file:///tmp/build/80754af9/send2trash_1607525499227/work.
- six @ file:///C:/ci/six_1605205426665/work.
- sktime==0.6.0.
- statsmodels==0.12.2.

- terminado==0.9.4.
- testpath @ file:///home/ktietz/src/ci/testpath_1611930608132/work.
- threadpoolctl==2.1.0.
- tornado @ file:///C:/ci/tornado_1606935947090/work.
- traitlets @ file:///home/ktietz/src/ci/traitlets_1611929699868/work.
- typing-extensions @ file:///home/ktietz/src/ci_mi/typing_extensions_16128082-09620/work.
- wcwidth @ file:///tmp/build/80754af9/wcwidth_1593447189090/work.
- webencodings==0.5.1.
- wincertstore==0.2.
- zipp @ file:///tmp/build/80754af9/zipp_1615904174917/work.