Árboles de decisión y Series de tiempo

21 de diciembre de 2009

Tesis de Maestría en Ingeniería Matemática

Facultad de Ingeniería, UDELAR

Ariel Roche

Director de Tesis: Dr. Badih Ghattas

Co-Tutor: Dr. Marco Scavino

Tribunal:

Dr. Ricardo Fraiman

Dr. Eduardo Grampin

Dr. Ernesto Mordecki

Dr. Gonzalo Perera

:
Agradezco por su dedicación, compañerismo e impulso, a Badih, Juan, Gonzalo, Marco, Anita, Mathias, Laurita, Paola, Pepe y Cacha.
También por su paciencia, a Dieguin, Paulita y a mi Amor de siempre.
Un recuerdo especial a mi mamá.

RESUMEN. Dentro de los métodos enmarcados en el aprendizaje automático supervisado, muchos pueden adaptarse a los problemas que tratan con atributos en forma de series de tiempo. Se han desarrollado métodos específicos, que permiten captar mejor el factor temporal. Muchos de ellos, incluyen etapas de pre-procesamiento de los datos, que extraen nuevos atributos de las series para su posterior tratamiento mediante métodos tradicionales. Estos modelos suelen depender demasiado del problema particular y a veces también resultan difíciles de interpretar. Aquí nos propusimos desarrollar un algoritmo, específico para clasificación y regresión con atributos series de tiempo, sin tratamiento previo de los datos y de fácil interpretación. Implementamos una adaptación de CART, cambiando la forma de particionar los nodos, utilizando la medida DTW de similaridad entre series. Aplicamos el método a la base artificial CBF, ampliamente utilizada en el contexto de clusterización y clasificación de series de tiempo. También experimentamos en un problema de regresión, con datos reales de tráfico en redes de internet.

Índice general

Capítulo 1. Introducción 7

Capítulo 2. Aprendizaje automático 9
1. Modelo general 10
2. Función de pérdida, riesgo 10
3. Principio de minimización del riesgo empírico 12
4. Errores en la predicción 13
5. Estimación de la performance del predictor 14
6. Algunas técnicas del apendizaje automático 15

Capítulo 3. Árboles de Clasificación y Regresión 19
1. CART 19
2. Árboles de Clasificación 21
3. Árboles de Regresión 36

Capítulo 4. Aprendizaje automático y Series de tiempo 41
1. Análisis de datos funcionales 41
2. Dynamic Time Warping 42
3. Complejidad del algoritmo de cálculo de la DTW 50
4. Clasificación con atributos series de tiempo 55

Capítulo 5. Árboles de decisión con atributos series de tiempo 59
1. Principios del método AST 59
2. Reducción en el número de particiones utilizadas 62
3. Agregación de modelos. Bagging. 63

Capítulo 6. Experimentación 65
1. Base CBF 65
2. Implementación 66
3. Resultados del modelo AST 68
4. Resultados del modelo AST-Bagging 73
5. AST en regresión - Tráfico en redes de internet 74

Capítulo 7. Conclusiones y perspectivas 79

Bibliografía 81

CAPíTULO 1

Introducción

En muchas áreas como la ingeniería, medicina, biología, etc., aparecen problemas de clasificación o regresión, donde los atributos de los datos tienen la forma de series de tiempo. Desde el punto de vista del aprendizaje automático, cada punto de cada serie puede considerarse como un atributo del individuo, para poder de esa manera aplicar cualquiera de los métodos tradicionales. El inconveniente radica en que de esta forma, puede suceder que no se logre captar adecuadamente, el efecto temporal en la estructura de los datos.

Es así, que se han desarrollado diversas metodologías específicas para dominios temporales. Muchas de ellas, proponen definir nuevos atributos por intermedio de la extracción de patrones o características de las series y posteriormente aplicar algún método tradicional a esas nuevas variables. En muchos casos, los modelos obtenidos son difíciles de interpretar y en otros resultan muy espécificos a la aplicación considerada.

Como los árboles de decisión suelen ser muy efectivos y de fácil interpretación en diversas situaciones, decidimos explorar la posibilidad de su utilización, en el contexto de atributos con características temporales. Siguiendo la estructura de CART [**3**, Breiman y otros, 1984], donde en cada nodo interno del árbol, se define una partición binaria, preguntando si el valor de uno de los atributos del individuo supera o no un determinado umbral, se puede trasladar esa idea y determinar una partición, planteándose si uno de los atributos serie de tiempo del individuo está "cerca" o no, de la correspondiente serie de tiempo de un individuo tomado como referencia.

Surge entonces la necesidad de utilizar una medida de similaridad entre series de tiempo. Encontramos en la literatura un consenso bastante amplio, en cuanto a la efectividad de la medida DTW introducida por [**46**, Sakoe y Chiba, 1978], especialmente cuando se quieren permitir ciertas deformaciones en el eje del tiempo. Si bien, esta medida fue originalmente desarrollada para su utilización en áreas específicas como el reconocimiento de voz, más tarde se generalizaron extensamente sus aplicaciones a otras áreas. Resulta fundamental tener en cuenta, que la

aplicación reiterada de los algoritmos para calcular la DTW, en muchas aplicaciones reales, suele tener una alta demanda computacional, por lo que en muchos casos se hace necesario implementar técnicas que permitan superar este inconveniente.

Los métodos que desarrollamos en base a las ideas anteriores fueron aplicados, en el caso de clasificación, a una base de datos sintéticos (CBF), que intenta simular determinados fenómenos temporales. Como esta base ha sido utilizada en muchos trabajos a los cuales hacemos referencia, también nos resultó útil a los efectos de comparar nuestros resultados. En el caso de regresión, los aplicamos a un problema de tráfico en redes de internet.

El resto del trabajo se organiza de la siguiente manera, en el Capítulo 2 se hace una revisión básica de conocimientos del aprendizaje automático. En el Capítulo 3, se analiza con cierto detalle el método CART de árboles de decisión, desarrollado en [**3**, Breiman y otros, 1984]. En el Capítulo 4, hablamos de series de tiempo en el contexto del aprendizaje automático, de la medida de similaridad DTW y de su utilización en algunas aplicaciones al problema de clasificación con atributos series de tiempo. En el Capítulo 5, damos los detalles de nuestro método para árboles de decisión con atributos series de tiempo. En el Capítulo 6, presentamos resultados de experimentos con la base de datos CBF, con los datos de tráfico en redes y los comparamos con otros métodos. En el Capítulo 7, sacamos algunas conclusiones y dejamos planteados ciertos temas para profundizar e investigar.

CAPíTULO 2

Aprendizaje automático

El aprendizaje automático (AA) consta de un conjunto de técnicas capaces de ayudar a resolver problemas de modelización en distintas áreas como ser, la biología, economía, informática, metereología, telecomunicaciones, etc. Algunos ejemplos:

- Predecir si un paciente tiene o no una determinada enfermedad, basándose en variables clínicas y demográficas.
- Establecer agrupamientos entre países, en función de algunas variables económicas.
- Asignar a un correo electrónico un puntaje del 0 al 10, vinculado a su nivel de "spam", teniendo en cuenta algunas características del mismo.

El AA, además de predecir una determinada variable, nos puede brindar una mejor comprensión del fenómeno de estudio desde el punto de vista de la causalidad, por ejemplo, estableciendo relaciones y jerarquías entre las variables involucradas. Otra ventaja es que pueden manejarse grandes bases de datos.

En muchas situaciones, el problema consta de algunas variables que deseamos predecir (cantidad de dólares exportados, presencia o no de una enfermedad) y otras variables que suponemos pueden explicarlas (tipo de cambio, edad del paciente). En esos casos hablaremos de **aprendizaje supervisado**. Otras veces, tenemos un conjunto de datos en los cuales no se determina una variable a explicar y el objetivo es organizar los datos de alguna manera, lo llamamos **aprendizaje no supervisado**. Un caso típico es el llamado *"clustering"*, que intenta determinar una partición de un conjunto de datos. Por ejemplo, supongamos que contamos con cierta información sobre una centena de países que le compran carne bovina al Uruguay y queremos agruparlos de algún modo, de manera de diseñar para cada grupo una estrategia particular de "marketing". Este trabajo se enmarca en el caso supervisado.

1. Modelo general

Sea un vector aleatorio (X, Y) donde:

- X es la llamada variables **de entrada** (también denominada independiente, explicativa, atributo), que toma valores en el espacio medible S_X.
- Y es la llamada variable **de salida** (también denominada dependiente, de respuesta), toma valores en el espacio medible S_Y.
- P es la probabilidad subyacente sobre el espacio de probabilidad donde está definido el vector aleatorio (X, Y).
- γ es la distribución conjunta del vector (X, Y), es decir

$$\gamma(A \times B) = P(X \in A, Y \in B) \quad \forall\, A \subset S_X \text{ y } B \subset S_Y$$

- $p(. \mid X)$ es la distribución condicional de Y dado X.
- π es la distribución marginal de X, es decir $\pi(A) = P(X \in A)$, con lo cual

$$\gamma(A \times B) = \int_A p(B \mid X)\, \pi(dx) = \int_A \int_B p(dy \mid X)\, \pi(dx)$$

En el problema de predicción se observa X y se trata de predecir el valor de Y por medio de $f(X)$ donde

$$f : S_X \to S_Y$$

es una función medible llamada **predictor**.

Cuando la variable Y es continua hablamos de problemas de **regresión** y cuando es categórica de **clasificación**.

2. Función de pérdida, riesgo

Para determinar la bondad de un predictor f definimos la **función de pérdida**

$$L : S_X \times S_Y \times S_Y \to \mathbb{R},$$

donde $L(x, u, y)$ representa la "pérdida" que significa, a partir de x, tomar $u = f(x)$ en lugar del verdadero valor y.

Luego se trata de elegir f de manera de minimizar el **riesgo** R_L definido como:

$$R_L(f) = E[L(X, f(X), Y)] = \int_{S_X} \int_{S_Y} L(x, f(x), y) \gamma(dx, dy).$$

Veamos como algunos problemas clásicos de la estadística, cuando se elige la función de pérdida adecuadamente, pueden verse como problemas de minimización del riesgo.

2.1. Problema de estimación de regresión

. Supongamos que estamos en el problema de regresión, S_Y es un espacio normado y $E\left[\|Y\|^2\right] < \infty$. Tomando como función de pérdida a

$$L(x,u,y) = \|u-y\|^2,$$

entonces

$$\begin{aligned} R_L(f) &= E\left[(f(X)-Y)^2\right] \\ &= \int_{S_X}\int_{S_Y}(f(x)-y)^2\,\gamma(dx,dy) \\ &= \int_{S_X}\left[\int_{S_Y}(f(x)-y)^2\,p(dy\mid x)\right]\pi(dx) \\ &= E_X E_{Y|X}\left[(f(X)-Y)^2\mid X\right], \end{aligned}$$

por lo cual para minimizar R_L es suficiente minimizar punto a punto

$$f(x) = \arg\min_c E_{Y|X}\left[(f(X)-c)^2\mid X\right].$$

Llegamos a que la solución que minimiza el riesgo es la llamada **función de regresión,**

$$f^*(x) = E\left[(Y\mid X=x)\right].$$

Por más detalles ver [**24**, Hastie y otros, 2001].

2.2. Reconocimiento de patrones

. El clásico problema de **reconocimiento de patrones,** formulado en los años 50 del siglo pasado, consiste en clasificar un objeto en k clases a partir de determinadas observaciones del mismo. Por ejemplo, en un estudio publicado en [**51**, Webb y Yohannes, 1999], se intenta determinar que hogares, de una población de Etiopía, son vulnerables a padecer hambre. En dicho estudio, el padrón 1 significa *vulnerable* y el 0 *no vulnerable***.** De cada hogar, se conocen una serie de variables como ser, rendimiento de las plantaciones de alimentos, cantidad de bueyes, ingreso por habitante, género del cabeza de familia, etc.

Formalmente, supongamos que el supervisor clasifica cada entrada X asignándole un valor $Y \in \{0,1,....,k-1\}$. Tomando como función de pérdida a

$$L(x,u,y) = \mathbf{1}_{\{u\neq y\}} = \begin{cases} 1 & \text{si } u \neq y \ \ \text{(clasifico mal)} \\ 0 & \text{si } u = y \ \ \text{(clasifico bien)} \end{cases}$$

entonces

$$R_L(f) = E\left[\mathbf{1}_{\{f(X)\neq Y\}}\right] = P\left(f(X)\neq Y\right).$$

Quiere decir que minimizar la probabilidad de clasificar mal, en este caso, consiste en minimizar el riego.

2.3. Estimación de densidades

. Supongamos que queremos estimar la densidad g de una determinada población. Si tomamos como función de pérdida

$$L(x, g(x), y) = -\ln g(x),$$

con lo cual

$$R_L(g) = E(L(X, g(X))) = \int -\ln g(x)\, g(x)\, dx$$

y el problema de estimar la densidad en L_1, se reduce a minimizar esta función de riesgo. Ver [**50**, Vapnik, 1998, pag 30-32].

3. Principio de minimización del riesgo empírico

En el aprendizaje supervisado, dada una muestra de entrenamiento

$$(X_1, Y_1), (X_2, Y_2), ..., (X_n, Y_n) \quad iid \sim \gamma,$$

buscamos una función que minimice el riesgo

$$R_L(f) = E(L(X, f(X), Y)) \quad \text{con} \quad f \in \mathcal{F},$$

donde $\mathcal{F}$ es una cierta familia de funciones. Como en general γ es desconocida, no podemos resolver el problema directamente.

Un criterio posible, es buscar en la familia de funciones $\mathcal{F}$, aquella que minimiza el llamado **riesgo empírico**

$$R_{L,n}(f) = \frac{1}{n}\sum_{i=1}^{n} L(X_i, f(X_i), Y_i),$$

que es un funcional construido a partir de los datos. Este método se conoce como **principio de minimización del riesgo empírico.** Llamaremos

$$\widehat{f_n} = \arg\min_{f\in\mathcal{F}} \{R_{L,n}(f)\}$$

a dicho predictor.

La idea de minimizar el riesgo empírico, en la construcción del predictor, fue desarrollada extensamente desde 1971 por Vapnik y Chervonenkis, ver [**50**, Vapnik, 1998].

Por ejemplo, volviendo al problema de regresión, donde

$$L(X, f(X), Y) = \|Y - f(X)\|^2,$$

obtenemos que el riesgo empírico es

$$R_{L,n}(f) = \frac{1}{n}\sum_{i=1}^{n}\left[Y_i - f(X_i)\right]^2.$$

Quiere decir que en este caso, buscar f que minimice el riesgo empírico, significa resolver el conocido problema de mínimos cuadrados.

4. Errores en la predicción

Como dijimos, $\widehat{f_n}$ es el predictor que efectivamente podemos calcular en la práctica. Denominemos f^* al mejor predictor entre todos los posibles y f^{**} al mejor entre todos los de una cierta clase de funciones $\mathcal{F}$ (ver Figura 1).

La perdida de performance, debida a la diferencia entre f^* y f^{**}, depende de la eleccción de $\mathcal{F}$, es decir del modelo escogido. Habitualmente se le llama **error de aproximación**. Si no se elige una clase adecuada $\mathcal{F}$, este error no se podrá disminuir, aunque mejoremos la calidad de la muestra. Sin embargo, la pérdida de performance debida a la diferencia entre $\widehat{f_n}$ y f^{**} sí, es de naturaleza estadística, por lo que se la llama **error de estimación**. Lo deseable sería que si tenemos una muestra muy grande ($n \to +\infty$), entonces $\widehat{f_n}$ tienda a f^{**}.

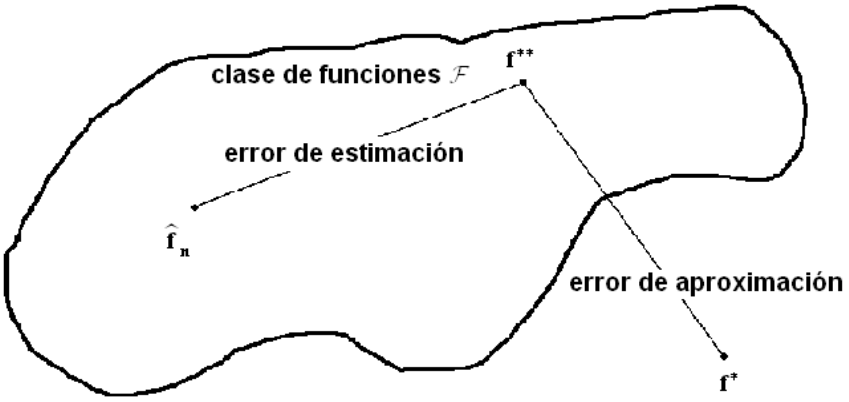

FIGURA 1. *Esquema representando los errores de aproximación y estimación que se producen al determinar el predictor*

Por ejemplo, en [**21**, Ghattas y Perera, 2004] se plantea el siguiente caso. Sea

$$R_{L,n}(f) = E_n\left[L(X, f(X), Y)\right],$$

donde E_n es la esperanza respecto a la distribución empirica F_n, definida como

$$F_n(C) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}_{\{(X_i,Y_i)\in C\}} \quad \forall\, C \subset S_X \times S_Y,\, C \text{ medible.}$$

Llamemos $L_{\mathcal{F}}$ a la clase de todas las funciones $g : S_X \times S_Y \to \mathbb{R}$ tales que existe $f \in \mathcal{F}$ para la cual $g(x,y) = L(x, f(x), y)$ para todo x, y. Diremos que $L_{\mathcal{F}}$ es una **clase Glivenko-Cantelli,** si se cumple una ley uniforme de grandes números, es decir

$$\lim_n \sup_{g\in L_{\mathcal{F}}} |E_n[g(X,Y)] - E[g(X,Y)]| = 0 \; c.s..$$

Si $\mathcal{F}$ es tal, que la clase de funciones asociada $L_{\mathcal{F}}$ es Glivenko-Cantelli, entonces

$$\widehat{f}_n \;\to f^{**} \; c.s.,$$

ya que

$$f^{**} = \arg\min_{f\in\mathcal{F}} E[L(X, f(X), Y)] = \arg\min_{g\in L_{\mathcal{F}}} E[g(X,Y)]$$

y

$$\widehat{f}_n = \arg\min_{f\in\mathcal{F}} E_n[L(X, f(X), Y)] = \arg\min_{g\in L_{\mathcal{F}}} E_n[g(X,Y)].$$

Para establecer, bajo qué condiciones se pueden lograr resultados más generales sobre esa convergencia, se ha desarrollado una vasta teoría, en la que se utilizan desigualdades exponenciales (Bernstein, Hoeffding, etc.) y la llamada **dimensión de Vapnik-Chervonenkis** de una clase de funciones, ver [**50**, Vapnik, 1998] y [**12**, Devroye, 1996]**.**

5. Estimación de la performance del predictor

La elección de la clase $\mathcal{F}$ es un compromiso entre los dos tipos de errores mencionados. Si tomamos una clase muy amplia de funciones, seguramente mejoraremos el error de aproximación, pero eso será en desmedro del error de estimación. Veamos el siguiente ejemplo. Supongamos que tomamos como familia $\mathcal{F}$ a todas las funciones medibles de S_X en S_Y. A partir de la **muestra de entrenamiento**

$$(X_1, Y_1), (X_2, Y_2), ..., (X_n, Y_n) \quad iid \sim \gamma,$$

eligimos el predictor de la siguiente manera:

$$\widehat{f}_n(x) = \begin{cases} Y_i & \text{si } x = X_i \\ 0 & \text{en otro caso} \end{cases}.$$

Si la función de pérdida es

$$L(x, u, y) = \mathbf{1}_{\{u\neq y_i\}},$$

entonces el riesgo empírico del predictor es

$$R_{L,n}\left(\widehat{f}_n\right) = \frac{1}{n}\sum_{i=1}^{n} L(X_i, \widehat{f}_n(X_i), Y_i) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}_{\left\{\widehat{f}_n(X_i)\neq Y_i\right\}} = 0.$$

Si tomamos como estimador de la performance de dicho predictor a $R_{L,n}\left(\widehat{f}_n\right)$, entonces $\widehat{f}_n$ obviamente resulta óptimo. En este caso se alcanza un ajuste perfecto a la muestra, pero cuando en la práctica se aplique el modelo obtenido a nuevos datos, el resultado puede resultar muy malo. El modelo sigue exactamente los datos de la muestra de entrenamiento, y eso desde el punto de vista estadístico deja mucho que desear, es el conocido **problema del sobre-ajuste**.

Una manera de detectar este tipo de problemas, es realizar la evaluación del predictor en base a otro conjunto de datos, llamado **muestra de prueba**

$$\left(X_1', Y_1'\right), \left(X_2', Y_2'\right), ..., \left(X_m', Y_m'\right) \quad iid \sim \gamma.$$

Esta muestra tiene la misma distribución que la muestra de entrenamiento, pero es independiente de ella. Luego, estimamos la performance del predictor con

$$R'_{L,n}(f) = \frac{1}{m}\sum_{i=1}^{m} L(X_i', f(X_i'), Y_i').$$

Otras técnicas estadísticas, utilizadas para estimar la performance del predictor, son la **validación cruzada** (*cross-validatión*)**,** introducida por M. Stone, [**49**, Stone, 1974], que emplearemos en las Subsecciones 2.6, 2.7 del Capítulo 3 y **el re-muestreo** (*bootstrap*)**,** desarrollada por B. Efron, [**13**, Efron, 1979].

6. Algunas técnicas del apendizaje automático

Son muchos los métodos que se enmarcan en el AA, buena parte de ellos comparten algunos principios. En particular, vamos a presentar tres de estos principios: particiones recursivas, modelos agregados y separación lineal, que incluyen a muchos de los métodos más conocidos, en particular a la mayoría de los que aparecen mencionados en este trabajo. Algunos de ellos suelen contar con algunas limitaciones, otros se destacan por presentar ventajas específicas, como permitir el cálculo de la importancia de las variables, tomar en cuenta interacciones, o permitir el manejo de datos missing.

6.1. Particiones recursivas

. La idea básica de este tipo de métodos, es crear una partición óptima en el espacio de los atributos y asignar un valor a la variable de respuesta en cada conjunto de la partición. Algunos de ellos son el algoritmo **ID3** [**36**, Quinlan, 1986], **C4.5** [**37**, Quinlan, 1993], **CHAID** [**28**, Kass, 1980] y **CART** [**3**, Breiman y otros, 1984]. Las diferencias, están en la forma en que construyen la partición óptima y en la manera de asignar los valores de las variables de salida. CART es uno de los primeros métodos no lineales y no paramétricos que ofrece un índice de importancia de las variables. Ha tenido una gran recepción y a partir de el se han desarrollado muchas extesiones en variadas direcciones:

- Salidas multivariadas continuas [**47**, Segal, 1992], salidas multivariadas discretas [**56**, Zhang, 1998].
- Salidas funcionales [**20**, Ghattas y Nerini (2006)], [**55**, Yu y Lambert (1999)].
- Particiones oblicuas [**34**, Murthy y otros, 1993].
- Particiones mas generales [**33**, Morimoto y otros, 2001].
- Predicción por modelos de regresion en la hojas.

6.2. Agregación de modelos

. En este caso, el principio básico consiste en construir varios modelos y luego combinarlos convenientemente para obtener uno nuevo. Existen varias extrategias que podríamos resumir en:

- Estimar diferentes modelos a partir de los datos disponibles, para luego agregarlos mediante una **regresión ridge**, obtenida a partir de las salidas de esos modelos. Un ejemplo es la denominada Stacked Regression [**5**, Breiman, 1996].
- Mediante re-muestreo de la muestra inicial, obtener un modelo para cada nueva muestra y al final decidir promediando, en el caso de regresión o haciendo voto mayoritario ponderado en clasificación. Es la idea básica en **Bagging** (**B***ootstrap* **Agg***regat***ing**) [**4**, Breiman, 1994], **Boosting** [**17**, Freund y Shapire, 1997] y **Bosques aleatorios** (*Random Forest*) [**6**, Breiman, 2001].

Estos métodos han demostrado tener una buena performance, debido a lo cual gozan de gran aceptación. Sin embargo, algunos de ellos no permiten su extensión directa a situaciones más complejas, por ejemplo boosting en al caso multi-clase. Otros inconvenientes son su mayor costo computacional y la dificultad para interpretar los resultados.

6.3. Separación lineal

. A pesar de que estas ideas son antiguas [**16**, Fisher, 1936], vuelven a surgir en la teoria del AA. El principio fundamental en estos métodos, es hacer una separación lineal de los datos en el espacio original, si es posible, o de lo contrario en uno más complejo. En este último caso, están comprendidos los llamados métodos Kernel, que ofrecen una alta flexibilidad y extensiones directas a varios tipos de conjuntos de datos. Tal vez, una de las aplicaciones más conocidas de este principio son las **Máquinas de Vectores de Soportes** (**S***upport* ***V****ector* ***M****achine,* **SVM**), [**9**, Cristiani y Shawe, 2004]. Estos métodos, no son muy claros en algunos contextos usuales, como salidas multi-clase o salidas multi-variadas discretas y continuas.

CAPíTULO 3

Árboles de Clasificación y Regresión

Los denominados árboles de decisión, constituyen uno de los métodos del aprendizaje inductivo supervisado más utilizados. Una de sus principales virtudes, es la sencillez de los modelos obtenidos. Dado un conjunto de ejemplos de entrenamiento, se construye una partición del espacio de entrada y se asigna a cada región un determinado modelo. Luego, dado un nuevo dato, a partir de los valores de las variables de entrada se determina una región y el predictor del modelo construido le asigna un valor a la variable de salida. Existen muchos métodos de árboles de decisión, como los introducidos por Quinlan, el algoritmo ID3 [**36**, Quinlan, 1986] y el C4.5 [**37**, Quinlan, 1993]. Nos concentraremos aquí en los llamados Árboles de Clasificación y Regresión, en ingles *Classification and Regression Trees* (CART), que deben su desarrollo a L. Breiman, J. Friedman, R. Olshen y C. Stone, autores del libro del mismo nombre, publicado en 1984 [**3**, Breiman y otros, 1984].

1. CART

Partimos de una muestra de entrenamiento

$$(X_1, Y_1), (X_2, Y_2), ..., (X_n, Y_n) \quad iid \sim \gamma, \tag{1.1}$$

donde cada $X_i = (X_i^1, ..., X_i^p)$ es un vector con p variables aleatorias, que pueden ser todas continuas, todas discretas o mezclas de ambas. Las variables Y_i son unidimensionales, discretas o continuas. Con la muestra de entrenamiento, construímos una estructura del tipo árbol en dos etapas bien diferenciadas, en la primera, determinamos el llamado **árbol maximal** y en la segunda, aplicamos un procedimiento denominado de **poda**.

Para construir el árbol maximal, comenzamos con toda la muestra en el **nodo raíz** y vamos obteniendo los **nodos interiores** por particiones sucesivas, mediante una cierta pregunta o regla que involucra a uno de los p atributos. Se trata de árboles binarios, por lo cual en función de la respuesta, cada nodo se parte en dos **nodos hijos**. Por convención, asignamos el nodo izquierdo al caso afirmativo y el derecho al contrario. Por último, se elige algún criterio de parada, para

saber cuando un nodo deja de partirse y queda constituyendo un nodo terminal llamado **hoja**.

Veamos el siguiente ejemplo. Supongamos una muestra de entrenamiento como en (1.1), con $p = 2$, y un árbol de 5 hojas como se representa en la Figura 1, donde las particiones se hacen en base a preguntas sobre los atributos X^1 y X^2 y los c_j son números reales. En la Figura 2, se aprecia como el espacio $\mathbb{R}^2$ queda partido en regiones R_i, con $i = 1, ..., 5$, donde cada R_i es un rectángulo de lados paralelos a los ejes.

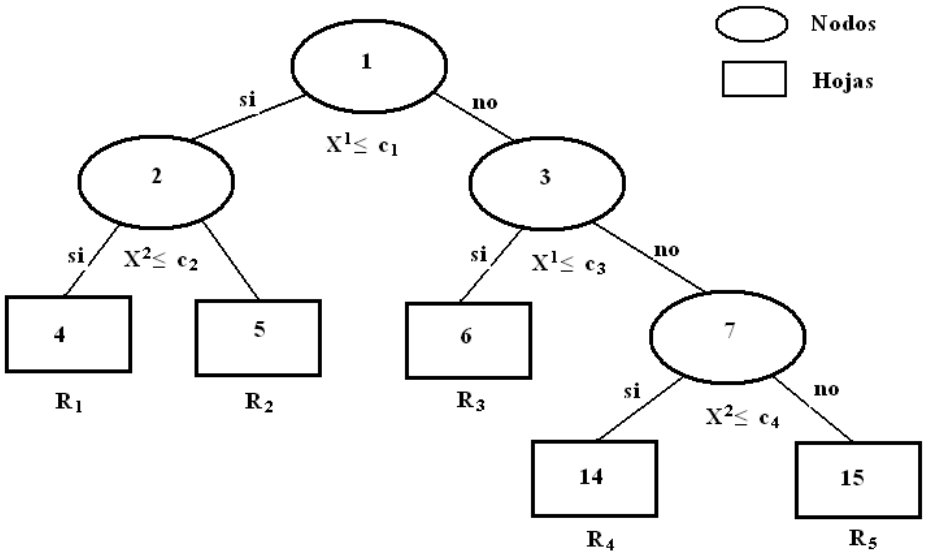

FIGURA 1. Ejemplo de un árbol de 5 hojas.

Una vez construido el árbol máximal, el predictor le asigna a cada región (**hoja**) un determinado valor:

$$E\left[Y \mid \left(X^1, X^2\right)\right] = \sum_{j=1}^{5} f_j\left(X^1, X^2\right) \mathbf{1}_{\{(X^1, X^2) \in R_j\}}$$

donde, si Y es continua (**árbol de regresión**) estimamos f_j por

$$\widehat{f_j} = \frac{\sum_{\{i:\left(X_i^1, X_i^2\right) \in R_j\}} Y_i}{card\left\{i : \left(X_i^1, X_i^2\right) \in R_j\right\}} \text{ (promedio de los } Y_i \text{ en la región } R_j \text{)}$$

y si Y es discreta (**árbol de clasificación**)

$$\widehat{f_j} = \text{la clase más frecuente en } R_j.$$

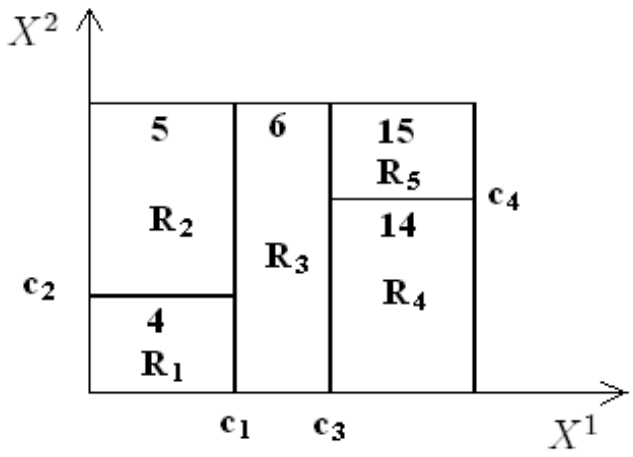

FIGURA 2. *Partición del espacio R^2, según el árbol de la Figura 1.*

Una de las características fundamentales de CART, es que luego de obtenido un árbol maximal se inicia una etapa de poda, en la cual se eliminan algunas de sus ramas. Este proceso, se explica con más detalle en la Sub-sección 2.7 de este Capítulo.

Comenzaremos por el desarrollo de los árboles de clasificación.

2. Árboles de Clasificación

Si bien, la estructura general y gran parte de los algorítmos de CART son similares para regresión y clasificación, preferimos verlos por separado, comenzando por los **árboles de clasificación,** que corresponden al caso en que Y toma valores en $\{1, ..., J\} \subset \mathbb{N}$.

2.1. Reglas de partición

. El objetivo fundamental, que nos planteamos al construir una partición, es optimizar la homogeneidad de las regiones resultantes. En cada nodo del árbol, nos proponemos aumentar la pureza de los dos nodos obtenidos.

Las reglas de partición en un nodo, dependen exclusivamente de los atributos, por lo cual son las mismas tanto para clasificación como para regresión. Para el caso de atributos discretos, supongamos que X^j toma valores en un conjunto finito $\{1, ..., H\} \subset \mathbb{N}$, entonces las reglas son de la forma

$$X^j \in C \text{ con } C \subset \{1, ..., H\}.$$

El número de todas las posibles reglas para cada atributo discreto X^j es

$$\frac{2^H - 2}{2} = 2^{H-1} - 1.$$

En el caso de atributos continuos, planteamos reglas del tipo

$$X^j \leq c, \text{ con } c \in \mathbb{R},$$

por lo cual las posibles reglas serían infinitas. Lo que hacemos en CART, es ordenar los valores que efectivamente toma cada atributo X^j, sobre la muestra de entrenamiento, elegir un punto intermedio m entre cada par de valores consecutivos y solamente considerar las reglas

$$X^j \leq m.$$

Por lo tanto, el número de todas las posibles reglas es a lo sumo $n-1$. En definitiva, en ambos casos las reglas son una cantidad finita y están perfectamente determinadas.

2.2. Criterio de partición de un nodo

. Entre todas las reglas posibles de partición de un nodo, debemos elegir la que mejor contribuya al aumento de la homogeneidad de sus dos hijos. Esto se logra, definiendo una medida de impureza sobre la variable de respuesta. Aquí sí, es fundamental tener en cuenta que estamos considerando que la variable Y es discreta.

Empecemos definiendo **función de impureza,** como una función ϕ definida sobre un conjunto de $J-$uplas $(p_1, ..., p_J) \in \mathbb{R}^J$, tales que

- $p_j \geq 0 \quad \forall\ j = 1, ..., J$
- $\sum_{j=1}^{J} p_j = 1,$

que verifica las siguientes propiedades:

- ϕ tiene un único máximo en $\left(\frac{1}{J}, ..., \frac{1}{J}\right)$,
- ϕ tiene mínimo 0 y solamente lo alcanza en los puntos de la forma

$$(1, 0, ..., 0), (0, 1, ..., 0), ..., (0, 0, ..., 1),$$

- ϕ es una función simétrica de $(p_1, ..., p_J)$.

Dada una función de impureza ϕ, podemos definir para cada nodo t de un árbol su **medida de impureza** $i(t)$, como

$$i(t) = \phi\left[p_1(t), ..., p_J(t)\right],$$

donde $p_j(t)$ es la probabilidad condicional de que un elemento pertenezca a la clase j, dado que pertenece al nodo t y puede estimarse en la práctica, como la proporción de elementos de clase j en el nodo t. Es

claro, a partir de estas definiciones, que la máxima impureza en el nodo t se da cuando todas las clases están igualmente representadas y la mínima se obtiene cuando en t hay casos de un único tipo.

Algunos de los criterios más utilizados en CART, como medida de impureza de un nodo, son:
la **medida de entropía**

$$i_{ent}(t) = -\sum_{j=1}^{J} p_j(t) \log p_j(t), \text{ definiendo } 0 \log 0 = 0,$$

y el **índice de Gini**

$$i_{Gini}(t) = \sum_{\substack{i,j=1 \\ i \neq j}}^{J} p_i(t)\, p_j(t) = 1 - \sum_{j=1}^{J} [p_j(t)]^2.$$

En el libro de Breiman [**3**, Breiman y otros, 1984, pag 38], se afirma que la elección de la medida de impureza más adecuada depende del problema y que el predictor construido, no parece ser muy sensible a dicha elección.

Supongamos entonces, que mediante una regla hemos partido al nodo t en dos, t_I (nodo izquierdo) y t_D (nodo derecho). Sea p_I la proporción de elementos del nodo t que caen en el hijo izquierdo y p_D la del derecho, ver Figura 3.

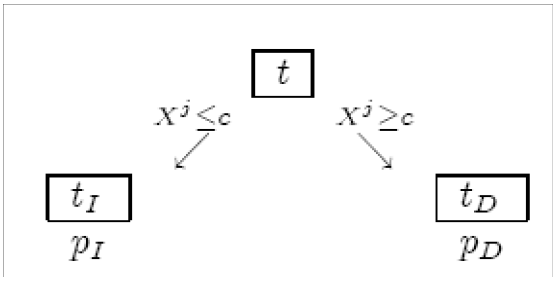

FIGURA 3. *Partición de un nodo, según si el atributo X^j supera o no al umbral c.*

Establecemos una medida de **bondad de una partición** s, para un nodo t, de la siguiente manera:

$$\Delta i(s,t) = i(t) - p_I i(t_I) - p_D i(t_D) \geq 0.$$

Es claro, que el aumento de la bondad depende de la disminución de la impureza, en los nodos hijos en relación al nodo padre. El criterio de

selección de la mejor partición s^* en el nodo t, consiste en elegir aquella que proporciona la mayor bondad

$$\Delta i\left(s^*,t\right)=\underset{s\in\psi}{\text{máx}}\left\{\Delta i\left(s,t\right)\right\},$$

donde ψ es el conjunto de todas las particiones posibles del nodo t.

2.3. Impureza del árbol

. Mediante las reglas de partición y comenzando desde el nodo raiz, se van partiendo sucesivamente los nodos. Una vez que termina el proceso de partición se obtiene un árbol, al cual nos va a interesar medirle la impureza, es decir cuantificar conjuntamente la impureza de todas sus hojas. Para eso, definimos la **impureza del árbol** A de la siguiente manera

$$I\left(A\right)=\sum_{t\in\widetilde{A}}i\left(t\right)p\left(t\right),$$

donde $\widetilde{A}$ es el conjunto de hojas del árbol A y $p\left(t\right)$ es la probabilidad de que un caso pertenezca al nodo t.

En [**3**, Breiman y otros, 1984, pag 32-33] los autores demuestran un resultado fundamental, **las selecciones sucesivas que maximizan** $\Delta i\left(s,t\right)$ **en cada nodo, son equivalentes a las que se realizarían con el fin de minimizar la impureza global del árbol**. Esto significa, que la estrategia de selección de la mejor división en cada nodo, lleva a la solución óptima, en términos del árbol final.

2.4. Regla de asignación de clases

. Una vez determinado que un nodo es terminal (hoja), corresponde asignarle una clase. La manera más habitual de hacerlo es por el método del **voto mayoritario**, que consiste en asignarle al nodo t la clase j^* si

$$p_{j^*}\left(t\right)=\underset{j=1,....,J}{\text{máx}}p_j\left(t\right)$$

y en caso de empate hacer un sorteo.

2.5. Reglas de parada

. Hasta ahora no hemos explicado como detener el proceso de partición, es decir cuándo declarar a un nodo lo suficientemente puro y que no se justifique partirlo. De la forma como resolvamos este problema, dependerá el tamaño del árbol construido. Un árbol muy grande, generará sobre-ajuste al conjunto de entrenamiento y uno muy pequeño, puede contribuir a que se pierda parte importante de la estructura de los datos.

Un posible criterio de parada, podría ser el de utilizar las medidas de impureza definidas anteriormente, declarando que un nodo es terminal si la disminución de impureza no supera determinado umbral. Umbrales muy bajos, generan árboles muy grandes, con los inconvenientes ya comentados. En cambio, umbrales mayores, pueden implicar que un nodo no se divida, cuando en realidad, una posterior partición de sus descendientes, sí está en condiciones de generar un buen decrecimiento de impureza.

Otros criterio utilizados son, evitar partir un nodo si la cantidad de elementos es menor que un determinado umbral, por ejemplo menor que 5, o si algunos de los dos nodos, que resultan de la partición óptima, no supera un umbral.

Por lo expuesto, el tamaño del árbol es un parámetro de ajuste fundamental para el modelo, por lo que debería escogerse en función de los datos (aprendiendo de los datos). La solución a la que llegan Breiman y demás autores, en [**3**, Breiman y otros, 1984], es primero construir un árbol llamado **maximal,** con la única condición de no permitir nodos con muy pocos elementos, para luego aplicarle un proceso denominado **poda**. Este consiste, en tomar el árbol maximal y sacarle aquellas ramas o sub-árboles que determinen beneficios muy pequeños, en lo respecta a la disminución de la impureza. Con este procedimiento se obtiene un sub-árbol, que permite para determinados nodos, que una de sus ramas permanezca y la otra se pode, al contrario de los criterios de parada anteriormente mencionados, en los cuales el efecto es el equivalente a podar simultáneamente ambas ramas. La construcción del árbol óptimo, la haremos a partir de un proceso de selección de sub-árboles, en la que interviene de manera fundamental el error asociado a cada uno de ellos. Por lo tanto, antes de pasar a explicar el método, vamos a ver como estimar esos errores.

2.6. Estimación del error de clasificación

. Supongamos que a partir de la muestra de entrenamiento construimos un árbol A. Como siempre, desconocemos el verdadero error de clasificación $R^*(A)$, por lo cual trabajamos con algún estimador $R(A)$ que surge de los datos. Pasemos a definir los estimadores usados más frecuentemente.

- **Estimador del error por resustitución, $\mathbf{R}^{res}$.**

Definimos, el estimador por resustitución del error global de clasificación del árbol A, como

$$R^{res}(A) = \sum_{t \in \widetilde{A}} \underbrace{r(t)\, p(t)}_{R^{res}(t)},$$

donde $p(t)$ es la proporción de elementos del nodo t sobre el total de ejemplos considerados y $r(t) = (1 - p_{j^*}(t))$ es el estimador por resustitución del error de clasificación en el nodo t (proporción de elementos mal clasificados). En [**3**, Breiman y otros, 1984, pag 95] se demuestra, que si se construye un árbol A' a partir de otro A partiendo alguno de sus nodos, entones

$$R^{res}(A') \leq R^{res}(A),$$

o de otra manera

$$R^{res}(t) \geq R^{res}(t_I) + R^{res}(t_D)$$

para todo nodo no terminal t. Esto quiere decir, que a medida que el árbol crece, el error por resustitución disminuye. Observemos que llegado al caso extremo, en que cada hoja tenga un solo elemento, se cumple que

$$R^{res}(A) = \sum_{t \in \widetilde{A}} \underbrace{\left(1 - \overbrace{p_{j^*}(t)}^{=1}\right)}_{=0} p(t) = 0.$$

En la práctica, la utilización de este estimador tiene como inconveniente el mencionado problema del sobre-aprendizaje o sobre-ajuste, que tratamos en la Sección 5 del Capítulo 2. El predictor construido, se ajusta demasiado al conjunto de entrenamiento, tiene un error por resustitución muy bajo, pero poco realista respecto a una evaluación sobre nuevos datos. Los errores que definimos a continuación intentan corregir este problema, de forma que el predictor construido tenga una estructura más general, menos ajustada a una muestra en particular. Se les suele llamar **estimadores honestos**.

- **Estimador por muestra de prueba, $\mathbf{R}^{mp}$.**

Partimos la muestra inicial M en dos subconjuntos:

M_1 **muestra de entrenamiento**

M_2 **muestra de prueba**,

de forma de utilizar M_1 para la construcción del árbol y M_2 para su evaluación. Luego, obtenemos el estimador por muestra de prueba repitiendo los cálculos del estimador por resustitución, pero tomando los datos del conjunto M_2 en lugar de la muestra completa. Los tamaños de ambas muestras no tienen porque ser iguales, por ejemplo pueden

ser $card\,(M_1) = \frac{2}{3} card\,(M)$ y $card\,(M_2) = \frac{1}{3} card\,(M)$, u otros. Cuando el tamaño de la muestra no es lo suficientemente grande, este estimador no es el más adecuado, por lo cual se puede optar por el siguiente.

- **Estimador por validación cruzada, $\mathbf{R}^{vc}$.**

En este caso, partimos la muestra M en V conjuntos del mismo tamaño (o lo más aproximadamente posible) $M_1, M_2, ..., M_V$. Construímos un primer árbol A_1, tomando $M^1 = M - M_1$ como muestra de entrenamiento y M_1 como muestra de prueba para su evaluación. De esta manera obtenemos el error

$$R_1^{vc}\,(A_1) = \sum_{t \in \widetilde{A}_1} r(t) p\,(t)\,.$$

Después, repetimos el mismo procedimiento, tomando como muestra de prueba M_2 y como muestra de entrenamiento $M^2 = M - M_2$. Así, sucesivamente calculamos $R_2^{vc}\,(A_2)\,, ..., R_V^{vc}\,(A_V)$, para finalmente definir el estimador por validación cruzada como

$$R^{vc}\,(A) = \frac{1}{V} \sum_{v=1}^{V} R_v^{vc}\,(A_v)\,.$$

Observamos que es muy frecuente utilizar $V = 10$, aunque también se suelen tomar otros valores como 5 o 20.

2.7. Elección del árbol óptimo, poda por mínimo costo-complejidad

. Como dijimos anteriormente, árboles muy grandes generan problemas de sobre-ajuste, además de que son más difíciles de interpretar. Es así que en [**3**, Breiman y otros, 1984] se plantea la siguiente metodología, que dividimos en dos etapas, para reducir el tamaño del árbol manteniendo una buena performance del clasificador.

2.7.1. ***Construcción de una secuencia de árboles podados***

. Comenzamos construyendo el llamado árbol máximal, $A_{\text{máx}}$, aplicando las reglas de partición y los criterios de parada mencionados anteriormente.

Llamamos **rama** de un árbol, al conjunto formado por un nodo y todos sus descendientes. Decimos que A' es un **sub-árbol** de A, y lo notamos $A' \prec A$, si A' se obtiene podando A, es decir eliminando alguna de sus ramas. La idea es, a partir de $A_{\text{máx}}$, obtener por podas sucesivas una secuencia anidada de sub-árboles, decrecientes en cantidad de hojas

$$A_K \prec A_{K-1} \prec .... \prec A_1 = A_{\text{máx}}.$$

Para esto tendremos en cuenta, que si bien al podar un árbol podemos perder bondad (aumentar el error), en la medida que esta pérdida sea pequeña vale la pena hacerlo, pues se gana en simplicidad, es decir se obtiene un árbol menos complejo.
Antes de continuar hagamos algunas definiciones.

- Llamamos **complejidad de un árbol** A, a su cantidad de hojas, $card\left(\widetilde{A}\right)$,
- **medida de costo-complejidad** asociada al árbol A, $R_\alpha(A)$, a

  $$R_\alpha(A) = R(A) + \alpha \times card\left(\widetilde{A}\right) \textbf{ con } \alpha \in \mathbb{R} \text{ y } \alpha \geq 0$$

- y **parámetro de complejidad** a α**.**

Nos planteamos eliminar, mediante podas, algunas ramas del árbol de partida, de manera de reducir $R_\alpha(A)$. Observemos que $R_\alpha(A)$ es una combinación lineal entre el error o costo del árbol y su complejidad (tamaño). Valores altos de α penalizan árboles con muchas hojas. Si por ejemplo, $\alpha \approx +\infty$, llegamos al absurdo de que el mejor árbol es el que tiene una sola hoja, aunque sea el que tiene peor costo $R(A)$. En el caso extremo contrario, un valor de $\alpha = 0$ no tiene en cuenta el tamaño, se queda con el que tiene menor $R(A)$, es decir con el maximal. Se trata de encontrar un compromiso entre bondad y complejidad. Consideremos el siguiente ejemplo, representado en la Figura 4, que intenta ilustrar esta situación.

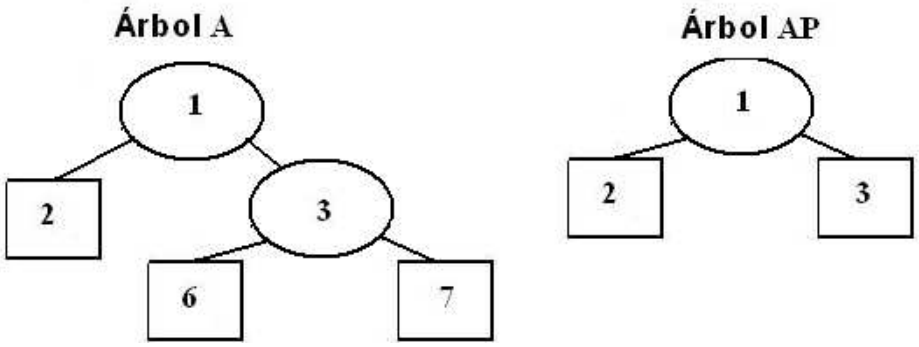

FIGURA 4. *A la izquierda, el árbol A de 3 hojas. A la derecha, el árbol AP de 2 hojas, obtenido a partir de A, por poda del nodo 3.*

El árbol A tiene complejidad 3 y al podarlo obtenemos el sub-árbol AP con complejidad 2. Supongamos que $\alpha = 0{,}1$ y que los errores valen $R(A) = 0{,}24$ y $R(AP) = 0{,}37$. Es claro que AP es menos complejo

que A, pero a su vez tiene asociado un error mayor. Medido en costo-complejidad:

$$R_\alpha(A) = R(A) + \alpha \times card\left(\widetilde{A}\right) = 0{,}24 + 0{,}1 \times 3 = 0{,}54$$

y

$$R_\alpha(AP) = R(AP) + \alpha \times card\left(\widetilde{AP}\right) = 0{,}37 + 0{,}1 \times 2 = 0{,}57$$

Como $R_\alpha(A) < R_\alpha(AP)$ no hacemos la poda. Si ahora, elegimos $\alpha = 0{,}15$ obtenemos

$$R_\alpha(A) = R(A) + \alpha \times card\left(\widetilde{A}\right) = 0{,}24 + 0{,}15 \times 3 = 0{,}69$$

y

$$R_\alpha(AP) = R(AP) + \alpha \times card\left(\widetilde{AP}\right) = 0{,}37 + 0{,}15 \times 2 = 0{,}67,$$

con lo cual, al ser $R_\alpha(A) > R_\alpha(AP)$, sí haríamos la poda.

El procedimiento entonces, parte de un árbol maximal, $A_1 = A_{\text{máx}}$, tomando $\alpha = 0$. Sabemos que para cualquier poda, se obtiene un sub-árbol podado AP que cumple

$$R(A) < R(AP)$$

y por lo tanto

$$R_{\alpha=0}(A) < R_{\alpha=0}(AP).$$

A medida que α crece, pero permanece pequeño, se mantiene esa relación. Al seguir incrementándose podría pasar, como en el ejemplo, que se invierta la relación. En otras palabras, la disminución en complejidad hace que el costo-complejidad mejore (disminuya), aunque el error sea mayor en el árbol podado. Se trata, de encontrar el menor valor de α para el cual existe un sub-arbol con mejor costo-complejidad, dicho de otro modo, encontar el menor valor de α y la rama más débil para podarla. A ese nuevo sub-árbol lo llamamos A_2 y al valor del parámetro α_2. Repitiendo sucesivamente el procedimiento, al final se encuentra la mencionada secuencia de árboles

$$A_K \prec A_{K-1} \prec .... \prec A_1 = A_{\text{máx}},$$

de la cual tendremos que seleccionar uno. Cada árbol de la secuencia tiene el menor costo $R(A)$, entre todos lo de su mismo tamaño. La descripción detallada del método se encuentra en [**3**, Breiman y otros, 1984, pag 66-71].

2.7.2. ***Elección del mejor árbol podado***

. De todos los árboles de la secuencia, nos quedamos con el que tiene asociado el menor error estimado R, es decir elegimos A_{K_0} si

$$R\left(A_{K_0}\right) = \min_K \left\{R\left(A_K\right)\right\}.$$

Como mencionamos en la Sub-sección anterior, es conveniente usar estimadores honestos en lugar del estimador por resustitución.

En el caso del estimador por muestra de prueba, se construye el árbol maximal y los sucesivos sub-árboles con la parte de la muestra destinada al entrenamiento y luego se calculan los errores $R\left(A_K\right)$ con la muestra de prueba, a los efectos de seleccionar el árbol óptimo.

A menos de que el tamaño de la muestra sea muy grande, es más conveniente usar el estimador de validación cruzada. Para eso, procedemos a partir la muestra M en V partes, como hicimos en la Subsección 2.6. De esta manera, se obtienen las sub-muestras

$$M^v = M - M_v \text{ con } v = 1, ..., V.$$

Para cada una de las sub-muestras M^v, sea A_K^v el árbol con mínimo costo-complejidad para el parámetro α_K. Luego, construimos con toda la muestra M, la secuencia de árboles $\{A_K\}$ y de parámetros $\{\alpha_K\}$. Siendo $\alpha_{K'} = \sqrt{\alpha_K \alpha_{K+1}}$, definimos $R^{vc}\left(A_K\right)$, como la proporción de elementos mal clasificados del total de la muestra M, donde cada elemento perteneciente a la muestra M_v lo clasificamos con el árbol $A_{K'}^v$, construido con la muestra M^v. Por último, elegimos de la secuencia $\{A_K\}$, el árbol A_{K_0} tal que

$$R^{vc}\left(A_{K_0}\right) = \min_K \left\{R^{vc}\left(A_K\right)\right\}.$$

2.8. La regla 1-SE

. La determinación del árbol óptimo predictor, tiene el inconveniente de la inestabilidad de los estimadores utilizados. Incluso, depende de cómo se efectúe la partición de la muestra inicial, tanto en el caso de R^{mp} como de R^{vc}. Podemos calibrar la variabilidad del estimador a partir de su error estándar. Si por ejemplo, consideramos $R = R^{mp}$, entonces el error estándar vale

$$SE\left[R^{mp}\left(A_K\right)\right] = \left\{\frac{R^{mp}\left(A_K\right)\left[1 - R^{mp}\left(A_K\right)\right]}{card\left(M_2\right)}\right\}^{\frac{1}{2}}$$

(recordemos que M_2 es la muestra de prueba).

En [**3**, Breiman y otros, 1984, pag 78-79], se afirma que en general el estimador R^{vc} se comporta como lo muestran las Figuras 5 y 6, como función de $card\left(\widetilde{A}_K\right)$ al principio decrece, para luego de un largo valle

volver a crecer. El mínimo, $R^{vc}(A_{K_0})$, se encuentra en ese valle, donde $R^{vc}(A_K)$ es casi constante, a menos de pequeñas variaciones en el rango de $\pm 1\ SE[R^{vc}]$. Entonces se propone el siguiente criterio, llamada **regla 1 SE,** que consiste en elegir como árbol óptimo al A_{K_1}, donde K_1 es el máximo K que verifica

$$R(A_{K_1}) \leq R(A_{K_0}) + SE(R(A_{K_0})).$$

En definitiva, eligimos el árbol más simple, entre todos los que tienen un error similar a A_{K_0}, como manera de mejorar la estabilidad del árbol predictor.

En la tabla de la Figura 5, se muestra una secuencia de sub-árboles $\{A_K : k = 1, ..., 17\}$ con sus respectivos valores de complejidad, errores R^{vc} y R^{res} y parámetro de complejidad, correspondiente al ejemplo mencionado en la Página 11, [**51**, Webb y Yohannes, 1999, Ejemplo 1]. En este caso, A_{11} sería el árbol elegido por el criterio de menor costo-complejidad, si utilizamos el estimador por validación cruzada. De todos los árboles que cumplen que sus errores R^{vc} son menores que $0{,}603 + 0{,}057$, nos quedamos con el más simple A_{12}, que tiene 8 nodos (ver Figuras 5 y 6).

También se observa en este ejemplo, que el error por resustitución decrece a medida que la complejidad crece, la elección del sub-árbol por el criterio de mínimo R^{res} nos llevaría a elegir el árbol maximal A_1, que resulta inconveniente como dijimos antes, por ser un modelo más complejo y sobre-ajustado a la muestra de entreamiento.

2.9. Importancia de las variables

. La construcción de un método de inducción como CART, no solamente tiene como objetivo la predicción, también permite analizar las relaciones entre las variables intervinientes. En particular, puede ser de interés, saber cuál de las variables de entrada inside más en la de salida. Una manera de hacerlo, es contabilizar en cuántas reglas de partición de los nodos interviene cada variable. Pero lo anterior no resulta un criterio satisfactorio, ya que puede pasar que una variable no aparezca en ninguna regla de partición y sin embargo juegue un rol fundamental, a tal punto, que si construyéramos de nuevo el árbol, eliminando alguna variable, sí, aparecería muchas veces y además el nuevo árbol podría resultar un buen clasificador, con precisión casi igual al original. Para buscar una mejor evaluación, de la importancia de cada variable, vamos a definir el concepto de partición sustituta.

K	$card\left(\widetilde{A_K}\right)$	$R^{vc}(A_K) \pm SE$	$R^{res}(A_K)$	α
1	32	0.704 ± 0.060	0.145	0.000
8	16	0.639 ± 0.058	0.244	0.008
9	14	0.635 ± 0.058	0.276	0.008
10	12	0.632 ± 0.058	0.310	0.008
11*	11	0.603 ± 0.057	0.332	0.011
12**	8	0.634 ± 0.058	0.430	0.016
13	7	0.668 ± 0.059	0.464	0.017
14	5	0.687 ± 0.059	0.540	0.019
15	3	0.700 ± 0.058	0.619	0.020
16	2	0.729 ± 0.048	0.696	0.038
17	1	1.000 ± 0.000	1.000	0.152

FIGURA 5. *La tabla corresponde a una secuencia de sub-árboles, a la cual se le aplica el método de poda por costo-complejidad. Datos reales, tomados de* [**51**, Webb y Yohannes, 1999, Ejemplo 1].

2.9.1. Partición sustituta

. Supongamos que hemos partido al nodo t, en los nodos t_I y t_d, mediante la partición óptima s^*. Dada una variable X^j, sea ψ_j el conjunto de todas las particiones de t en las que interviene X^j y $\overline{\psi}_j$ el conjunto de todas las particiones complementarias de ψ_j. Llamemos t'_I y t'_D a los nodos que resultan de aplicar una partición $s_j \in \psi_j \cup \overline{\psi}_j$.

Definimos, el estimador de la probabilidad de que las particiones s^* y s_j asignen un elemento del nodo t al nodo izquierdo como

$$p_{II}\left(s^*, s_j\right) = \frac{p\left(t_I \cap t'_I\right)}{p\left(t\right)},$$

al nodo derecho

$$p_{DD}\left(s^*, s_j\right) = \frac{p\left(t_D \cap \ t'_D\right)}{p\left(t\right)}$$

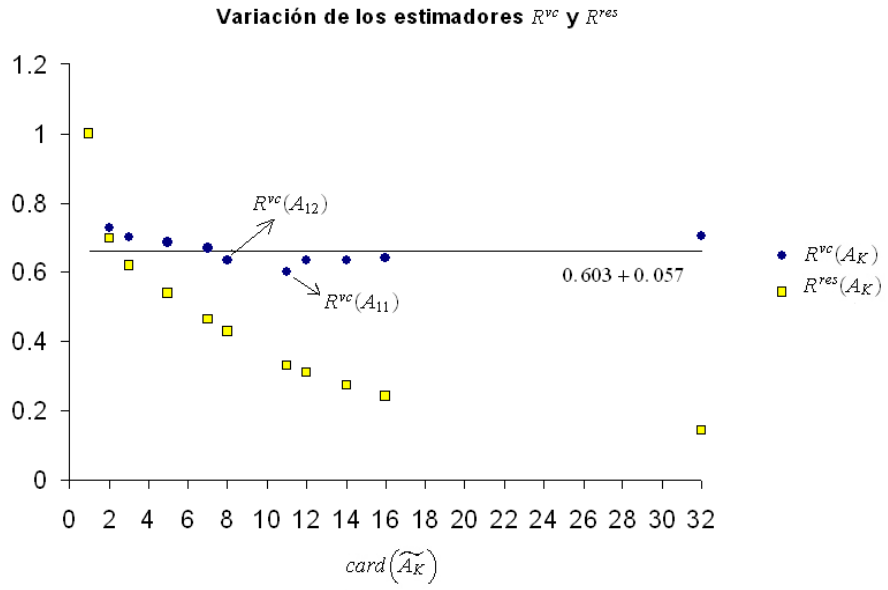

FIGURA 6. *Gráfico correspondiente a la tabla de la Figura 5.*

y el estimador de la probabilidad de que la partición s_j prediga correctamente la partición s^* como

$$p(s^*, s_j) = p_{II}(s^*, s_j) + p_{DD}(s^*, s_j) = \frac{p(t_I \cap t'_I) + p(t_D \cap t'_D)}{p(t)}.$$

Por último, definimos **partición sustituta** de s^* sobre la variable X^j, a la partición que maximiza al estimador anterior, es decir, a la partición $\overline{s}_j$ que verifica:

$$p(s^*, \overline{s}_j) = \max_{s_j \in \psi_j \cup \overline{\psi}_j} p(s^*, s_j)$$

2.9.2. Medida de importancia de una variable

. En base al concepto anterior, una manera de cuantificar la importancia global de una variable en un árbol, es tener en cuenta su capacidad de sustituir a la partición óptima en cada uno de los nodos. Es así, que definimos la medida de importancia de la variable X_j como

$$\tau(X_j) = \sum_{t \notin \widetilde{A},\ t \in A} \Delta i(\overline{s}_j, t)$$

($\Delta i(s_j, t)$ definida como en la Sub-sección 2.2, Página 23).

Como lo que realmente importa, es medir la importancia relativa entre las variables, resulta útil usar una medida normalizada como

$$\tau_{Norm}(X_j) = 100\frac{\tau(X_j)}{\max\limits_j \{\tau(X_j)\}},$$

de forma de asignarle a la variable más importante el puntaje 100 y a las demás un valor entre 0 y 100.

El concepto de partición sustituta, también resulta útil, cuando se presentan **datos missing,** ver [**3**, Breiman, 1994, pag 142-146].

2.10. Consistencia

. Sería razonable, que cuando el tamaño de la muestra de aprendizaje, con la cual construimos el árbol de clasificación, tienda a infinito, entonces el error empírico de clasificación, converja al error de clasificación. Pasemos entonces a formalizar este problema y a presentar un par de teoremas de convergencia.

Veamos previamente algunas notaciones y definiciones:

- Sean, el vector aleatorio (X, Y) y el correspondiente espacio de probabilidad $(\Omega, \mathcal{A}, P)$.
- $X = (X^1, ..., X^p) \in \mathbb{R}^p$ y llamamos P_X a su distribución de probabilidad.
- Y toma valores en $\{1, ..., J\}$, con $J \in \mathbb{N} - \{0, 1\}$.
- $\forall\ A \in \mathcal{A}$, llamamos $\nu(A) = P(X \in A)$.
- $P(j \mid x) = P(Y = j \mid X = x)$.
- La funcion de riesgo es

$$R(f) = \int [1 - P(f(x) \mid x)] P_X(dx).$$

La **regla de clasificación de Bayes** f^*, consiste en tomar $f(x)$ como el menor valor de $i \in \{1, ..., J\}$, que minimiza $1 - P(i \mid x)$.

Si la muestra de aprendizaje es

$$(X_1, Y_1), (X_2, Y_2), ..., (X_n, Y_n) \quad iid$$

y $\widehat{P}_n$ es la distribucion empírica de X_k, con $1 \leq k \leq n$, definimos $\widehat{f}_n$, eligiendo $\widehat{f}_n(x)$ como el menor valor de $i \in \{1, ..., J\}$ que mimimiza $1 - P_n(i \mid x)$.

- Vamos a definir particiones de $\mathbb{R}^p$, donde los conjuntos son **policlases**. Un conjunto $B \subset \mathbb{R}^p$ es una policlase, si sus elementos $x = (x_1, ..., x_p)$ verifican al menos p_1 inecuaciones del

tipo $\sum_{i=1}^{p} b_j x_j \leq a$ ó $\sum_{i=1}^{p} b_j x_j < a$, donde $p_1 \geq p+1$, $b_1, ..., b_p, a$ son reales y alguno de los b_j es distinto de cero. Estas policlases tienen asociados p_1 hiperplanos. Un conjunto B se denomina **policlase de base,** si para cada inecuación, exactamente un b_j es distinto de cero. Sea $d(n)$ la cantidad de hiperplanos que dependen del tamaño de la muestra de aprendizaje.

- Sea $\Pi = \{B_l\}_{1 \leq l \leq L}$, una partición de $\mathbb{R}^p$, donde los B_l son policlases. Sea $\left\{\Pi^{(n)}\right\}_{n \geq 1}$, una sucesión no creciente de particiones de $\mathbb{R}^p$, $\Pi^{(n+1)} \leq \Pi^{(n)}$, en el sentido de que todas las policlases de $\Pi^{(n+1)}$ están incluidas en alguna policlase de $\Pi^{(n)}$. Llamamos $B(x)$ y $B^{(n)}(x)$, respectivamente a las policlases de Π y $\Pi^{(n)}$, que contienen a $x \in \mathbb{R}^p$.
- Sea $\delta(A)$, el diámetro de un conjunto $A \subset \mathbb{R}^p$, definido de la siguiente manera

$$\delta(A) = \sup\{\|u - v\|,\ u \in A,\ v \in A\}$$

Teorema ([**4**, Breiman y otros (1984)]**)**
Si se cumplen las siguientes condiciones:

1. k_n es una sucesión de enteros positivos tales que
$$\lim_{n \to +\infty} k_n = +\infty$$
2. $$\mathbf{1}_{\left\{\widehat{P}_n\left(B^{(n)}(x)\right) < k_n \frac{\ln n}{n}\right\}} \overset{c.s}{\to} 0 \text{ si } n \to +\infty$$
3. $$\delta\left(B^{(n)}(x)\right) \overset{P}{\to} 0 \text{ si } n \to +\infty$$

entonces

$$\lim_{n \to +\infty} E\left|R\left(\widehat{f}_n\right)\right| = R(f^*).$$

Teorema ([**32**, Lugosi y Nobel, 1996]**)**
Sea $\widehat{f}_n$, el predictor de clasificación del árbol construido sobre una partición $\Pi^{(n)}$, basada sobre al menos $d(n) - 1$ hiperplanos de $\mathbb{R}^p$. Si se cumple:

1. $$d(n) = o\left(\frac{n}{\ln n}\right)$$
2. Para $0 < M < \infty$, para todo $\eta > 0$ y para toda bola $S_M = \{x \in \mathbb{R}^p : \|x\| \leq M\}$ se verifica
$$\lim_{n \to +\infty} \nu\left[x : \delta\left(B^{(n)}(x) \cap S_M\right) > \eta\right] = 0 \quad c.s$$

entonces

$$\lim_{n\to+\infty} R\left(\widehat{f}_n\right) = R\left(f^*\right) \quad c.s.$$

- En particular, $\widehat{f}_n$ es fuertemente consistente, si la condición 2 se verifica y si cada clase de la partición $\Pi^{(n)}$ contiene al menos h_n elementos, con

$$\lim_{n\to+\infty} \frac{h_n}{\ln n} = +\infty$$

3. Árboles de Regresión

Partiendo del modelo general para CART, visto en la Sección 1, ahora tomamos la variable de respuesta Y perteneciendo al campo real. El procedimiento general es similar al caso de clasificación, por lo cual seremos menos exhautivos, procurando hacer hincapié en aquellos aspectos donde radican las principales diferencias.

El tipo de reglas, según las cuales se parten los nodos, son las mismas, ya que dependen exclusivamente de las variables de entrada.

En donde se presentan las principales diferencias, es en la definición del criterio de pureza u homogeneidad de los nodos. En este caso, a diferencia de clasificación, usaremos el mismo, tanto para la construcción del árbol, como para el procedimiento de poda.

3.1. Particiones y errores

. Tomemos, el vector aleatorio (X, Y) como lo planteamos en el modelo de la Sección 1, la variable $Y \in \mathbb{R}$, la función predictor f y el **error cuadrático medio** de f

$$R^*(f) = E\left[(Y - f(X))^2\right].$$

Ya vimos, en la Sub-sección 2.1 del Capítulo 2, que la función que minimiza $R^*(f)$ es

$$f(x) = E\left[(Y \mid X = x)\right].$$

Supongamos, que a partir de la muestra

$$(X_1, Y_1), (X_2, Y_2), ..., (X_n, Y_n) \quad iid \sim \gamma,$$

construímos el árbol A y sea f_A el predictor asociado a dicho árbol. Como es habitual, el verdadero valor del error $R^*(f_A)$ no lo conocemos, por lo cual trabajamos con algún estimador $R(f_A)$.

Llamamos, estimador por resustitución a

$$R^{res}(f_A) = R^{res}(A) = \frac{1}{n}\sum_{i=1}^{n}\left[Y_i - f_A(X_i)\right]^2.$$

Para hacer este error mínimo, asignamos a cada $f_A(X_i)$ el valor

$$\overline{Y_t} = \frac{1}{card(t)}\sum_{X_n \in t} Y_n, \tag{3.1}$$

donde t es la hoja para la cual $X_i \in t$ y $card(t)$ es la cantidad de individuos en la hoja t. Si denominamos

$$R(t) = \frac{1}{n}\sum_{X_i \in t}\left[Y_i - \overline{Y_t}\right]^2,$$

resulta

$$R^{res}(A) = \sum_{t \in \widetilde{A}} R(t).$$

También, de forma similar a clasificación, eligimos en el conjunto de todas las posibles particiones ψ, aquella que maximiza el incremento en la disminución de impureza, de los nodos hijos en relación al nodo padre. Como medida de impureza tomamos R. Más precisamente, elgimos la partición s^* del nodo t tal que

$$\Delta R(s^*, t) = \max_{s \in \psi}\left\{\Delta R(s,t)\right\},$$

donde

$$\Delta R(s,t) = R(t) - R(t_I) - R(t_D).$$

En clasificación no usamos el error por resustitución, como criterio de impureza para determinar las particiones, pues trae aparejado algunos inconvenientes, ver [**3**, Breiman y otros, 1984, pag 94-98]. En regresión, no tenemos ese tipo de problemas.

El criterio adoptado, separa naturalmente a un nodo, en un nodo hijo con los valores más altos de la variable de respuesta y otro con los valores más bajos.

Observación:
Si escribimos:

$$R(t) = \frac{1}{n}\sum_{X_i \in t}\left[Y_i - \overline{Y_t}\right]^2 = p(t)\, var(t),$$

donde como antes

$$p(t) = \frac{card(t)}{n}$$

es el error por resustitución de la probabilidad de que un elemento perteneca a un nodo t y

$$s^2(t) = \frac{1}{card(t)} \sum_{X_i \in t} \left[Y_i - \overline{Y_t}\right]^2$$

es la varianza muestral de los valores Y_i que caen en el nodo t. Entonces

$$\begin{aligned} \Delta R(s,t) &= R(t) - R(t_I) - R(t_D) \\ &= R(t) - p_I(t)\, s^2(t_I) - p_D(t)\, s^2(t_D) \\ &= R(t) - \left[p_I(t)\, s^2(t_I) + p_D(t)\, s^2(t_D)\right], \end{aligned}$$

por lo cual podemos interpretar, que el criterio elegido como bondad de la partición, corresponde a seleccionar aquella que maximiza la varianza muestral ponderada de ambos hijos

$$p_I(t)\, var(t_I) + p_D(t)\, var(t_D).$$

3.2. Poda por mínimo-costo-complejidad

. Al igual que en clasificación, generamos por medio de particiones un árbol maximal, dividiendo los nodos hasta que tengan una cierta cantidad mínima de elementos, por ejemplo 5. A continuación, aplicamos en forma análoga el método de poda por **mínimo-costo-complejidad,** a partir de la **medida costo-complejidad:**

$$R_\alpha(A) = R^{res}(A) + \alpha \times card\left(\widetilde{A}\right) \text{ con } \alpha \in \mathbb{R} \text{ y } \alpha \geq 0$$

Una vez obtenida la secuencia de sub-árboles $\{A_K\}$, si optamos por validación por muestra de prueba, calculamos

$$R^{mp}(A_K) = \frac{1}{card(M_2)} \sum_{(X_i,Y_i) \in M_2} \left[Y_i - \overline{Y_t}\right]^2,$$

donde $\overline{Y}$ es el valor promedio definido por la Ecuación (3.1) de la Página 37, a partir del árbol A_K generado por la muestra M^2. En el caso de elegir validación cruzada, calculamos

$$R^{vc}(A_K) = \frac{1}{card(M)} \sum_{v=1}^{V} \sum_{(X_i,Y_i) \in M_v} \left[Y_i - \overline{Y}_t^v\right]^2,$$

donde cada $\overline{Y}_t^v$ se obtiene a partir del árbol $A_{K'}^v$, construido con la muestra M^v.

3.3. Error relativo

. En el caso de los árboles de clasificación, el error de clasificación tiene una interpretación intuitiva. Esto no sucede en el caso del error cuadrático medio, ya que el valor de R^*, depende de la escala donde se mida la variable de respuesta Y. Surge entonces, la necesidad de normalizar esta medida de error, de manera de independizarla de la escala. Una manera de hacerlo es dividir por la varianza.

Definimos, el **error relativo cuadrático medio** de un estimador f, como

$$RE^*(f) = \frac{R^*(f)}{Var(Y)}$$

con

$$Var(Y) = E\left[(X - E(X))^2\right]$$

Naturalmente, tomaremos como estimador de $Var(Y)$ a la varianza muestral

$$R\left(\overline{Y}\right) = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \overline{Y}\right)^2,$$

con

$$\overline{Y} = \frac{1}{n}\sum_{i=1}^{n} Y_i\ .$$

Finalmente, definimos **estimador por resustitucion del error relativo cuadrático medio** a

$$RE^{res}(f) = \frac{R^{res}(f)}{R\left(\overline{Y}\right)},$$

estimador por muestra de prueba del error relativo cuadrático medio a

$$RE^{mp}(f) = \frac{R^{mp}(f)}{R^{mp}\left(\overline{Y}\right)},$$

y **estimador por validación cruzada del error relativo cuadrático medio** a

$$RE^{vc}(f) = \frac{R^{vc}(f)}{R\left(\overline{Y}\right)}.$$

CAPíTULO 4

Aprendizaje automático y Series de tiempo

En muchas situaciones, las bases de datos que se pretenden analizar estadísticamente, si bien se presentan de manera discreta, puede suponerse que corresponden a modelos continuos. Por ejemplo, supongamos que para el mes de octubre, se tienen mediciones horarias de la temperatura, en una cierta estación meteorológica. Podemos entonces considerar, que tenemos una muestra $X_1, ..., X_{31\times 24}$ y trabajar con esos datos sin tener en cuenta el orden, que en este caso significa perder el carácter temporal. Aquí, la naturaleza del fenómeno "temperatura", nos hace pensar que se trata de una variable, que en realidad, varía de manera continua a lo largo del tiempo. Hay diversas maneras de tener en cuenta este hecho, una de las cuales es intentar ajustar una curva, que corresponda a una función temperatura $X(t)$, a partir de las mediciones consideradas. Es decir, obtener lo que se llama una variable funcional, para luego aplicar técnicas específicas para ese tipo de variables. Para obtener las mencionadas curvas, se pueden aplicar métodos de interpolación, suavizado u otros métodos no paramétricos. En otros casos, los datos, pueden presentarse directamente en forma funcional, por ejemplo, en problemas que involucran funciones de densidad. Puede verse un ejemplo interesante, de árboles de regresión funcional para la clasificación de densidades, en [**20**, Ghattas y Nerini, 2006]. El estudio de estas técnicas se denomina **Análisis de Datos Funcionales (ADF)**.

1. Análisis de datos funcionales

En el **Análisis de Datos Funcionales**, la unidad básica de información es la función completa, más que un conjunto de valores (Ramsay-DAlzell, 1991). En el contexto multivariado, los datos vienen de la observación de la familia aleatoria $\{X(t_i)\}_{i=1,....,n}$. En el análisis funcional, se asume que estos mismos datos provienen de una familia continua $\{X(t) : t \in T \subset \mathbb{R}\}$. Las siguientes definiciones son importantes en este contexto [**15**, Ferrati y Vieu, 2006].

- Una variable aleatoria X, se llama **variable funcional**, si toma valores en un espacio infinito dimensional (espacio funcional). Una observación x de X se llama **dato funcional**.
- Un conjunto de **datos funcionales** $x_1, ..., x_n$, es la observación de n variables funcionales $X_1, ..., X_n$, con igual distribución que X.

Usualmente

$$X \in L^2(T) = \left\{ f : T \to \mathbb{R}, \text{ tal que } \int_T f^2(t)\, dt < \infty \right\},$$

donde en $L^2(T)$ se toma el producto interno usual

$$\langle f, g \rangle = \int_T f(t)\, g(t)\, dt.$$

"Desde el trabajo pionero de Deville [**11**, Deville, 1974] y más recientemente con el de Ramsay& Dalzell [**38**, Ramsay y Dalzell, 1991], la comunidad estadística ha estado interesada en el análisis de datos funcionales (ADF). Se han propuesto versiones funcionales para métodos estadísticos tradicionales como, entre otros, regresión [**7**, Cardot y otros, 1999], análisis de varianza [**10**, Cuevas y otros, 2004], modelo lineal generalizado [**14**, Escabias y otros, 2004] o componentes principales [**35**, Pezulli y Silverman, 1993]. Los conceptos básicos del ADF y algunas de las metodologías antes mencionadas se encuentran en [**39**, Ramsay y Silverman, 2005]".(Ver [**23**, Giraldo, 2007, Página 118]).

A diferencia del tratamiento que se hace en ADF, existen otras formas de proceder, cuando los datos aparecen en forma de series de tiempo. Podemos contemplar el carácter temporal, sin hacer la transformación a datos funcionales. En casos como clasificación o regresión con atributos en forma de series de tiempo, esto se puede lograr utilizando una medida de similaridad entre las series. Una de estas medidas, es la denominada Dinamic Time Warping (DTW).

2. Dynamic Time Warping

2.1. Introducción

. Si tenemos dos series de tiempo

$$A = (a_1, a_2, ..., a_n) \text{ e } B = (b_1, b_2, ..., b_n)$$

una manera natural de medir la similaridad entre ambas es a través de la distacia euclidiana

$$d\left(A,B\right)=\sqrt{\sum_{i=1}^{i=n}\left(a_i-b_i\right)^2},$$

o también de la distancia

$$d\left(A,B\right)=\sqrt{\sum_{i=1}^{i=n}\left|a_i-b_i\right|}.$$

Podríamos aplicar alguna de estas distancias, como medida de similaridad entre las series de tiempo. Un primer inconveniente, radica en la imposibilidad de comparar dos series de tiempo de distinto tamaño. Pero incluso, en el caso de series de igual largo, una dificultad que presentan las distancias mencionadas, radica en que son muy sensibles a distorsiones en el eje del tiempo, aun cuando estas sean pequeñas. Nos referimos a que si por ejemplo, dos señales son iguales, salvo por un desplazamiento en el tiempo, la distancia euclidiana las reconocería como distintas por su característica de alineamiento punto a punto. Lo anterior se puede apreciar gráficamente en la parte superior de la Figura 1, un alineamiento entre los puntos de ambas series, como se sugiere en la parte inferior de esa Figura, resulta intuitivamente más conveniente.

En este sentido, fue introducida la denominada **DTW** (**Dynamic Time Warping**), que algunos autores traducen al español como Distancia de Alineamiento Temporal no Lineal. Se reconoce su principal antecedente en el trabajo de Hiroaki Sakoe y Seibi Chiba en 1978 [**46**, Sakoe y Chiba, 1978], en el marco de la resolución de problemas vinculados al reconocimiento de voz. En dicho artículo, los autores proponen *"an optimun dynamic programming based time-normalization for spoken word recognition"*. Se considera que dos personas pueden pronunciar la misma frase, pero de manera distinta, utilizando distintos períodos de tiempo, tanto para pronunciar las palabras, como para efectuar las pausas entre ellas. Las dos representaciones tendrán globalmente la misma forma, pero con deformaciones a lo largo del eje del tiempo. La idea es eliminar esas diferencias temporales, permitiendo incluso la alineación entre un punto de una serie con varios de la otra.

Posteriormente, la técnica DTW fue introducida en la comunidad de "data mining", entre otros por [**1**, Berndt - Clifford, 1994]. En los últimos 15 años abundan sus aplicaciones en diferentes áreas como, bioinformática (análisis de series de tiempo de datos de expresión de

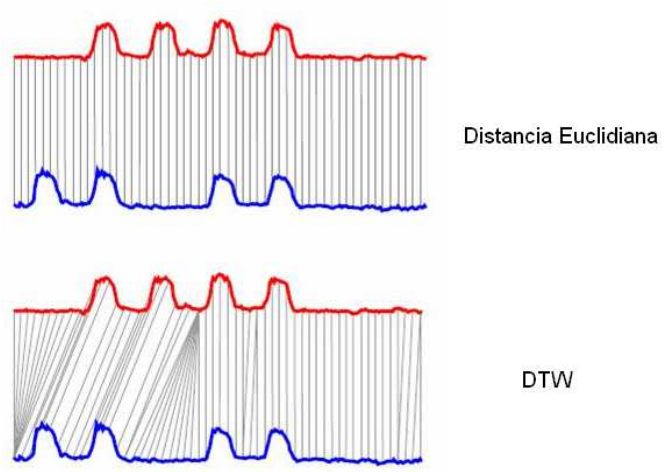

FIGURA 1. *Se representan dos series con la misma forma general. En la parte superior se muestra el alineamiento que produce la distancia euclidiana, el punto i-ésimo de la primera con el i-ésimo de la segunda. En la parte inferior se muestra un alineamiento no lineal, que permite una medida de similaridad más sofisticada. En ambos casos una de las series se desplazó en forma vertical para mejor visualización. La figura está tomada de* [**29**, Keogh, 2002].

ácido ribonucleico, recopilados de una gran cantidad de genes), ingeniería química (sincronización y monitoreo de procesamiento de lotes en polimerización), robótica (clustering de salidas de agentes sensoriales), medicina (correspondencia entre patrones en electrocardiogramas), ajuste biométrico de datos (firmas, huellas digitales), etc., ver [**29**, Keogh, 2002].

2.2. Definición de la medida de similaridad DTW

. Sean dos series de números reales A y B, de largo I y J respectivamente

$$\begin{aligned} A &= a_1, a_2, ..., a_i, ..., a_I \\ B &= b_1, b_2, ..., b_j, ..., b_J \end{aligned} \tag{2.1}$$

y consideremos (ver Figura 2) en el plano de ejes $\overrightarrow{i}$, $\overrightarrow{j}$, alineamientos entre los índices de ambas series, a partir de trayectorias F

$$F = c_1, c_2, ..., c_k, ..., c_K \tag{2.2}$$

de puntos

$$c_k = (i_k, j_k)\,,$$

con

$$\text{máx}(I, J) \leq K < I + J.$$

A estas trayectorias F, les exigimos que verifiquen ciertas restricciones que describiremos en la Sub-sección 2.3.

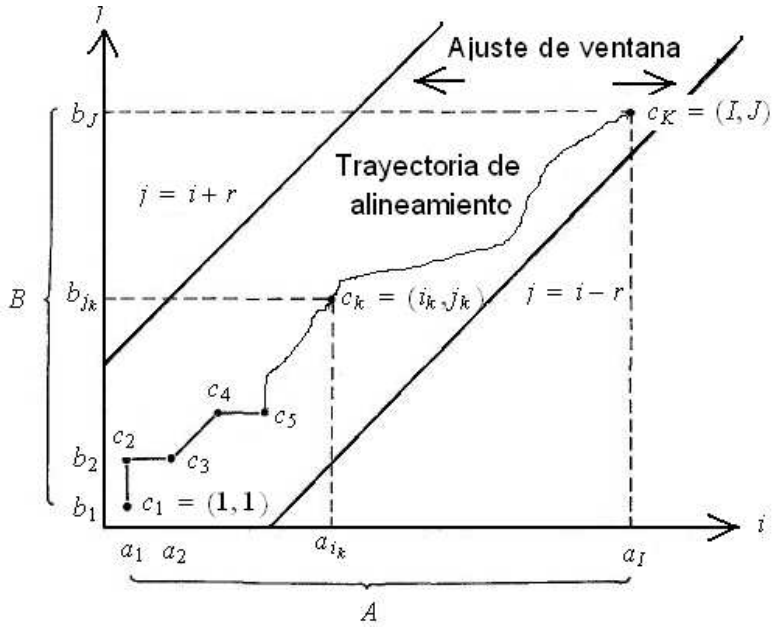

FIGURA 2. *Trayectoria de alineamiento de la DTW entre las series A y B, con ajuste de ventana de Sakoe-Chiba de tamaño r. Figura similar a* [**46**, Sakoe y Chiba, 1978, Figura 1].

Observemos, que si en particular

$$I = J = K \text{ e } i_k = j_k \;\; \forall\; k = 1, ..., K\;,$$

la trayectoria F corresponde al alineamiento punto a punto de la distancia euclidiana, es decir, la diagonal que une los puntos $(1, 1)$ y (I, J) en la Figura 2.

Para cada punto c_k, sea

$$d(c_k) = d\,(i_k, j_k) = \|a_{i_k} - b_{j_k}\| \tag{2.3}$$

la distancia entre los valores de ambas series que fueron alineados. Dada una trayectoria F, definimos la suma ponderada de dichas distancias,

$$S(F) = \sum_{k=1}^{k=K} d(c_k) w_k \quad \text{con } w_k \geq 0. \tag{2.4}$$

Volviendo al ejemplo, en donde las dos series representan la misma frase (padrón), pronunciada con variantes en el tiempo, en la medida que elegimos trayectorias F, que eliminan esas diferencias temporales, obtenemos sumas $S(F)$ menores. Una vez eliminadas todas las diferencias debidas a "deformaciones" en el tiempo, es razonable pensar que encontramos la trayectoria óptima F_{opt} y que la suma resultante $S(F_{opt})$ es la "verdadera" distancia entre ambas series.

Lo anterior, motiva la siguiente definición, de **medida de similaridad DTW** entre dos series A y B:

$$DTW(A,B) = \min_{F} \left\{ \frac{S(F)}{\sum_{k=1}^{k=K} w_k} \right\} \tag{2.5}$$

en donde el denominador, se introduce para compensar el efecto de la cantidad de puntos K que tiene cada trayectoria F .

2.3. Restricciones en las trayectorias de alineamiento

. A las trayectorias F le vamos a imponer ciertas condiciones, de manera que las deformaciones temporales en el alineamiento, queden sujetas a algunas restricciones:

- Condiciones de monotonía:

$$i_{k-1} \leq i_k \;\text{ y }\; j_{k-1} \leq j_k$$

- Condiciones de continuidad:

$$i_k - i_{k-1} \leq 1 \;\text{ y }\; j_k - j_{k-1} \leq 1$$

  Observemos, que a consecuencia de estas dos condiciones, el punto c_{k-1} solamente puede tomar tres valores:

$$c_{k-1} = \begin{cases} (i_k, j_k - 1) \\ (i_k - 1, j_k - 1) \\ (i_k - 1, j_k) \end{cases} \tag{2.6}$$

- Condiciones de borde:

$$i_1 = 1,\ j_1 = 1,\ i_K = I,\ j_K = J$$

- Condiciones de ajuste de ventana:

$$|i_k - j_k| \leq r \quad \text{con } r \text{ entero positivo} \tag{2.7}$$

Este tipo de condiciónes, se imponen como forma de impedir excesivas deformaciones temporales. Más adelante, en la Subsección 3.1, veremos que también nos van a permitir mejorar la velocidad de los algoritmos en el cálculo de la DTW. La ventana introducida en [**46**, Sakoe y Chiba, 1978], corresponde a la condición (2.7) y puede visualizarse en la Figura 2. También pueden considerarse otro tipo de ventanas, ver Figura 5.

- Condiciones sobre la pendiente:
Se trata de que la pendiente de la trayectoria, no sea ni muy suave ni muy empinada, de manera de impedir que un tramo corto de una de las series se corresponda con uno largo de la otra. La condición es, que si el punto c_k se mueve m-veces consecutivas en la dirección de uno de los ejes, entonces deberá moverse n-veces en la dirección de la diagonal, antes de volver a moverse en la misma dirección del eje anterior, ver Figura 3. La manera de medir la intensidad de esta restricción, es a través del parámetro $p = n/m$. Cuanto mas grande es p, mayor es la restricción. En particular, si $p = 0$ no hay restricción y si $p = \infty$ la trayectoria está restringida a moverse sobre la diagonal, es decir que no se permite deformación alguna.

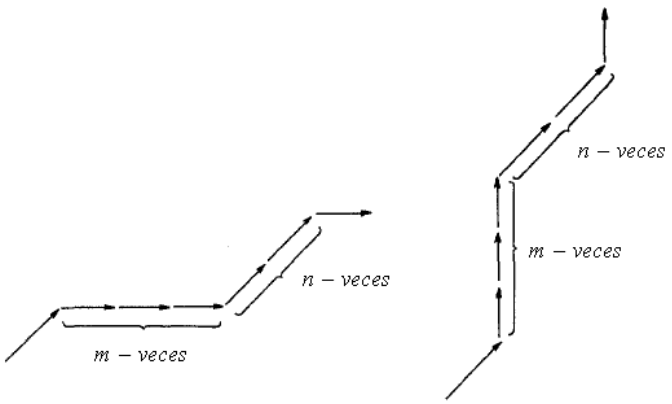

FIGURA 3. *Ejemplo de restricción en la pendiente para* $p = \frac{2}{3}$. *Tres movimientos del punto* c_k *en la dircción del eje vertical u horizontal, obliga a efectuar al menos dos movimientos en la dirección de la diagonal. Figura similar a* [**46**, Sakoe y Chiba, 1978, Figura 2].

2.4. Sobre los coeficientes de ponderación

. Volviendo a la definición de la medida DTW, y tomando en la expresión (2.5), el denominador

$$N = \sum_{k=1}^{k=K} w_k,$$

llamado **coeficiente de normalización,** en forma independiente a la trayectoria elegida, obtenemos

$$DTW\,(A,B) = \frac{1}{N} \min_{F} \left\{ \sum_{k=1}^{k=K} d(c_k) w_k \right\}. \tag{2.8}$$

La efectiva obtención del mínimo en (2.8) es muy compleja. Las siguientes formas, de definir los coeficientes de ponderación, permiten la aplicación de técnicas de programación dinámica, que simplifican sensiblemente el problema.

- Forma simétrica:

  $$w_k \overset{def}{=} (i_k - i_{k-1}) + (j_k - j_{k-1}) \Rightarrow N = I + J$$

  y se cumple que,

  $$\text{si } c_{k-1} = \begin{cases} (i_k, j_k - 1) \\ (i_k - 1, j_k - 1) \\ (i_k - 1, j_k) \end{cases} \Rightarrow w_k = \begin{cases} 1 \\ 2 \\ 1 \end{cases}.$$

- Forma asimétrica

  $$w_k = i_k - i_{k-1} \Rightarrow N = I$$

  y se cumple que,

  $$\text{si } c_{k-1} = \begin{cases} (i_k, j_k - 1) \\ (i_k - 1, j_k - 1) \\ (i_k - 1, j_k) \end{cases} \Rightarrow w_k = \begin{cases} 0 \\ 1 \\ 1 \end{cases}.$$

  (también podría ser $w_k = j_k - j_{k-1}$)

Observemos, que en el primer caso la DTW es simétrica:

$$DTW\,(A,B) = DTW\,(B,A)$$

y en el segundo no. Además, en el caso asimétrico notamos que en la expresión (2.8) no son tenidas en cuenta las distancias $d(c_k)$, cuando las trayectorias son verticales, ya que el correspondiente coeficiente w_k vale cero. En [**46**, Sakoe - Chiba, 1978], se afirma que por este motivo la performance de la forma simétrica es superior. De todas maneras se aclara, que si las restricciones sobre la pendiente son estrictas, estas

diferencias se reducen, al no permitirse excesivos movimientos en el sentido vertical.

2.5. Algoritmos de programación dinámica

. En la literatura consultada, acerca de clusterización y clasificación con atributos series de tiempo utilizando la DTW, encontramos que la mayoría de los autores, ([**29**, Keogh, 2002], [**8**, Chu y otros, 2002], [**42**, Rodriguez - Alonso, 2004], [**48**, Stan - Chan, 2007], [**52**, Xi y otros, 2006], [**53**, Yamada y otros, 2003]), trabajan con un modelo más simple que el dado por la expresión (2.8). Solamente aplican las restricciones vistas de monotonía, continuidad, borde y eventualmente de ventana. Además, toman todos los coeficientes de ponderación iguales a 1, con lo cual se obtiene

$$DTW\left(A,B\right)=\frac{1}{K}\min_{F}\left\{\sum_{k=1}^{k=K}d(c_k)\right\}, \tag{2.9}$$

o sin normalizar

$$DTW\left(A,B\right)=\min_{F}\left\{\sum_{k=1}^{k=K}d(c_k)\right\}. \tag{2.10}$$

Para este último caso, veamos como implementar un algoritmo de programación dinámica. Supongamos que tenemos como en (2.1) dos series de tiempo A y B y queremos calcular la $DTW\left(A,B\right)$, definida por la expresión (2.10). El procedimiento de determinar todas las trayectorias F, calcular $\sum_{k=1}^{k=K}d(c_k)$ para cada una de ellas y al final hallar el mínimo es muy ineficiente. En lugar de eso, utilizaremos un argumento de programación dinámica.

Comenzamos por definir la llamada **matriz de costos** D, de dimensión $I\times J$, cuyas entradas $D\left(i,j\right)$ quedan definidas a partir de las sub-series

$$\begin{aligned} A' &= a_1,a_2,....,a_i \quad i=1,....,I \\ B' &= b_1,b_2,....,b_j \quad j=1,....,J \end{aligned} \tag{2.11}$$

de la siguiente manera:

$$D\left(i,j\right)=DTW\left(A',B'\right). \tag{2.12}$$

Observemos, que a consecuencia de las restricciones de monotonía y continuidad impuestas a las trayectorias, resumidas en (2.6), podemos calcular cada elemento $D\left(i,j\right)$ de la matriz a partir de sus elementos

adyacentes $D(i-1,j)$, $D(i,j-1)$, $D(i,j)$ y de la distancia $d(i,j)$ entre a_i y b_j, mediante la siguiente fórmula de recurrencia:
(2.13)

$$D(i,j) = d(i,j) + \text{mín}\{D(i-1,j), D(i,j-1), D(i-1,j-1)\}.$$

La manera de calcular la matriz D, es ir llenando cada columna de abajo hacia arriba, comenzando por la columna de más a la izquierda y continuando hacia la derecha, ver Figura 4. Al terminar este proceso y por la propia definición (2.12), obtenemos la medida de similaridad buscada

$$DTW(A,B) = D(I,J).$$

A su vez, a partir de la matriz de costos podemos determinar la trayectoria óptima

$$F_{opt} = \widetilde{c}_1, \widetilde{c}_2, ..., \widetilde{c}_k, ..., \widetilde{c}_K.$$

En este caso, comenzamos por el punto $\widetilde{c}_K = (I,J)$. En el paso k, para el punto $\widetilde{c}_k = (i_k, j_k)$, buscamos en cuál de los 3 adyacentes ubicados a la izquierda y abajo $((i_k-1,j_k),(i_k,j_k-1),(i_k-1,j_k-1))$, D es mínimo. Así vamos construyendo la trayectoria hasta llegar al $\widetilde{c}_1 = (1,1)$.

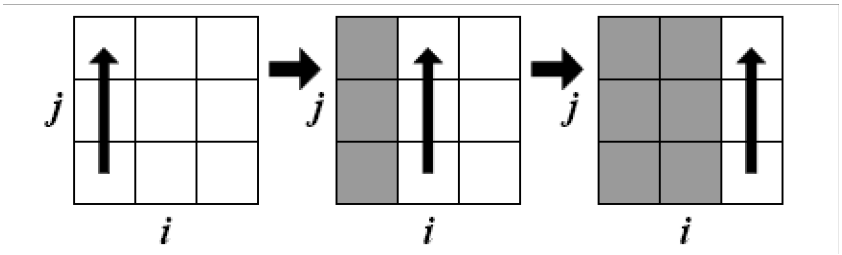

FIGURA 4. *Orden según el cual se calcula la matriz de costos. La figura está tomada de* [**48**, Stan y Chan, 2007, Figura 3].

3. Complejidad del algoritmo de cálculo de la DTW

Se deduce de la construcción del algoritmo, que su complejidad espacial y temporal es $O(I \times J)$ (ya que se necesita llenar todos los campos de la matriz de costos). En el caso particular que $I = J$, la complejidad es $O(I^2)$. La complejidad cuadrática en el espacio, podría ser un importante inconveniente para series muy largas, debido a los requerimientos de memoria. Si lo que se desea calcular es solamente la DTW, el problema puede resolverse mediante la implementación de un

algoritmo de complejidad lineal. En efecto, debido a que en el procedimiento descripto, en cada paso k solo se necesitan las columnas k y $k-1$, podemos ir desechando las anteriores. Sin embargo, si necesitamos la trayectoria del alineamiento óptimo, lo anterior no es posible, necesitamos mantener almacenada toda la matriz, ya que el proceso se inicia en el punto (I, J).

Por otra parte, en muchas aplicaciones tenemos una cantidad de series n y necesitamos calcular la DTW entre todas ellas, la complejidad total pasa a ser de $O(I^2 \times n^2)$. Se hace necesario por lo tanto, buscar procedimientos que permitan acelerar los cálculos cuando aparece involucrada la DTW, veamos algunos.

3.1. Restricciones en las trayectorias

. En la Sub-sección 2.3 hablamos de las condiciones de ajuste de ventana. En [**46**, Sakoe y Chiba, 1978], se introducen ventanas conocidas

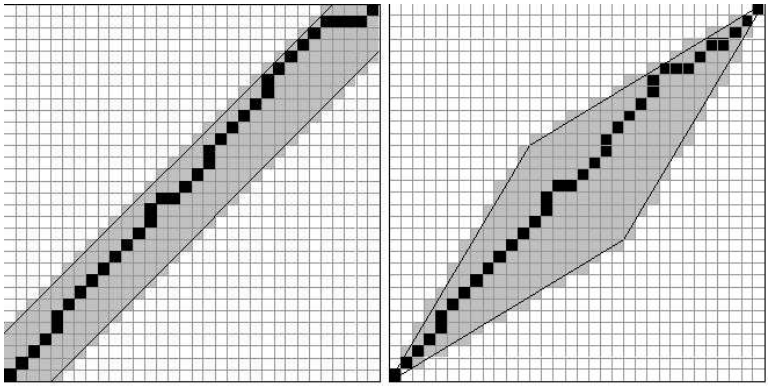

FIGURA 5. *A la izquierda: Banda de Sakoe-Chiba. A la derecha: Paralelogramo de Itakura.*

como "Bandas de Sakoe-Chiba", ver condición (2.7). Otras ventanas usadas frecuentemente son las denominadas "Paralelogramo de Itakura" [**25**, Itakura (1975)]. Podemos ver ejemplos de ambas condiciones en la Figura 5. En estos casos, los algoritmos son más rápidos, pues no se necesita calcular toda la matriz de costos, solamente los elementos correspondientes a las zonas sombreadas, aunque obviamente, la trayectoria y la correspondiente DTW encontradas no son las óptimas. Esto deja de ser un inconveniente, cuando las trayectorias óptimas se encuentran cerca de la diagonal. Por el contrario, cuando las deformaciónes en el tiempo son importantes, las DTW encontradas pueden alejarse de las verdaderas.

3.2. Representación de las series en forma reducida

. Otra línea de trabajo, consiste en calcular la DTW sobre una representación reducida de las series. Una manera elemental de hacerlo es partir la serie en intervalos iguales, para luego tomar como representación reducida, la serie formada por los promedios de los valores de la serie original en cada intervalo. En particular, sea la serie

$$A = a_1, a_2, ..., a_i, ..., a_I$$

y supongamos por simplicidad que $I = 2^u$, con $u \in \mathbb{N}$. Tendremos distintas representaciones

$$A^l = a_1, a_2, ..., a_i, ..., a_{2^l} \quad \forall\, l \in \{0, 1, ..., u\} \tag{3.1}$$

Por ejemplo, si $l = 2$ la representación reducida constará de 4 valores.

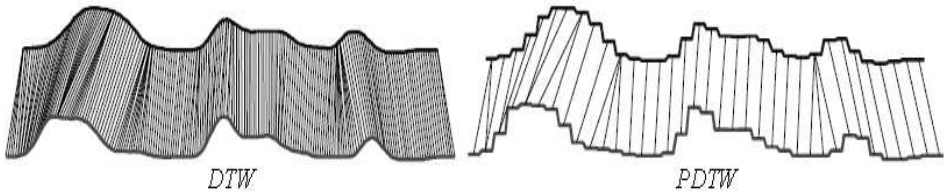

FIGURA 6. *A la izquierda, la alineación producida por la DTW. A la derecha, la alineación producida por la PDTW sobre las representaciones reducidas de las series originales. Figura tomada de* [**8**, Chu y otros, 2002].

En [**8**, Chu y otros, 2002], se propone una representación reducida de este tipo, denominada PPA (Piece Aggregate Aproximation) y un algoritmo que da una aproximación de la medida DTW, la llamada medida PDTW, que resulta de calcular la DTW entre las representaciones reducidas. Los autores reportan, que obtienen una importante reducción en la velocidad de cálculo en problemas de clasificación y clusterización, con poca pérdida de precisión. En la Figura 6 se muestra un caso, donde claramente se aprecia como el alineamiento de la PTDW, en las representaciónes reducidas, es muy similar al de la DTW en las series originales. En el mencionado trabajo se plantea un algoritmo llamado IDDTW (Iterative Deepening dynamic Time Warping), mediante el cual, dado un nivel preestablecido de tolerancia para falsos disímiles y mediante una fase de entrenamiento, se determina cual es el nivel de reducción óptimo $\widetilde{l}$ sobre el cual trabajar.

3.3. Limitación en las veces que se calcula la DTW

. Otros métodos proponen, en lugar de acelerar el algoritmo de la DTW, limitar la cantidad de veces que se requiere su cálculo en un determinado problema. Por ejemplo, en [**29**, Keogh, 2002] se utilizan las llamadas "lower-bounding measures" (LB). Supongamos que tenemos una medida de similaridad LB, entre las series A y B, tal que

$$LB(A,B) \leq DTW(A,B) \ \forall \ A \text{ y } B \tag{3.2}$$

y que además se puede calcular a una velocidad muy superior a la DTW. Si el problema de estudio, consiste en encontrar dentro de un conjunto de n series, aquella que sea más similar a la serie dada A, segun el criterio de la DTW, el autor propone una estrategia que se resume en el siguiente pseudo código:

```
1.    mejor_hasta_ahora = ∞
2.    para i = 1 hasta n
3.        Distancia_LB = LB(A, B_i)
4.        si Distancia_LB < mejor_hasta_ahora
5.            verdadera_distancia = DTW(A, B_i)
6.            si verdadera_distancia < mejor_hasta_ahora
7.                mejor_hasta_ahora = verdadera_distancia
8.                indice_mejor_serie = i
9.            fin si
10.       fin si
11.   fin para
```

De esta manera, se evita hacer el calculo de la DTW en los n casos. Para que este método realmente funcione, es necesario que además de cumplir la desigualdad (3.2), las medidas propuestas aproximen adecuadamente la DTW. En el artículo se propone una medida LB, denominada LB_Keogh, con las características mencionadas. Además de probar la desigualdad (3.2), compara empiricamente la medida LB_Keogh con otras dos medidas LB introducidas en [**30**, Kim y Park (2001)] y [**54**, Yi, Jagadish y Faloutsos (2001)], por medio de 32 bases de datos distintas. El autor afirma, que según sus conocimientos, son las únicas medidas LB disponibles para DTW y que LB_Keogh resulta superior, en términos de calidad de aproximación a la verdadera DTW y de cantidad de veces que se evita calcular la DTW.

3.4. El método FastDTW

. En un artículo más reciente [**48**, Stan y Chan, 2007], los autores afirman que las técnicas mencionadas para acelerar la DTW, no logran cambiar la complejidad cuadrática en el tamaño I de las series,

solamente la reducen un factor r, con $r << I$. A su vez, y tomando alguna de esas mismas ideas, plantean un nuevo algoritmo y demuestran teóricamente que su complejidad es lineal en el largo de las series. También muestran su buen rendimiento en cuanto a la precisión, evaluando múltiples bases de datos.

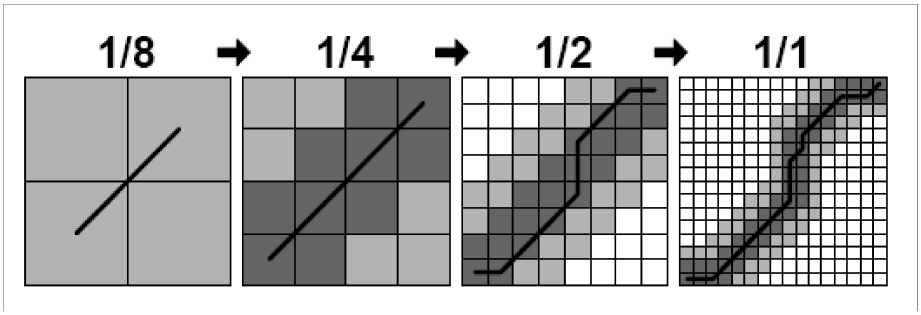

FIGURA 7. *Cuatro etapas del método Fast-DTW de Stan y Chan. Figura tomada de* [**48**, Stan y Chan, 2007].

Dadas las series A y B, el método llamado $FastDTW$ comienza por obtener representaciones reducidas A^l y B^l para un l "chico", como explicamos antes y mostramos en la Figura 6. Para estas representaciones reducidas se calcula la trayectoria óptima correspondiente a la $DTW\left(A^l, B^l\right)$. A continuación, se pasa a la siguientes representaciones A^{l+1} y B^{l+1} y se proyecta la trayectoria encontrada en este nuevo cuadro. En el ejemplo de la Figura 7, sería pasar del primer cuadro de la izquierda al segundo. Luego, se busca la trayectoria óptima según la DTW para este nuevo nivel de reducción, pero restringiendo la búsqueda a las celdas adyacentes a la trayectoria proyectada, (ver celdas sombreadas oscuras en la Figura 7). Así sucesivamente, hasta llegar a la resolución máxima ($2^u = I$), donde tomamos como aproximación de la verdadera $DTW(A, B)$ la $FastDTW\,(A, B)$, resultante de calcular la DTW restringida a las celdas adyacentes a la última proyección (cuadro de más a la derecha en la Figura 7). Como manera de aumentar las posibilidades de encontrar el camino óptimo, se toman en cada paso, otra cantidad s (radio) de celdas adyacentes a ambos lados de la trayectoria proyectada. En la Figura 7, las celdas sombreadas suavemente corresponden a un radio $s = 1$.

Los autores comparan la performance del método $FastDTW\,(A, B)$, con dos de las técnicas que mencionamos antes, a) bandas de Sakoe-Chiba y b) representación reducida de las series, sobre un amplio espectro de bases de datos. Reportan mejoras en la precisión, que van

desde 51 veces mejor que a) y 143 que b) para $s = 0$, hasta 3 veces mejor que a) y 15 que b) para $s = 30$. En cuanto a la velocidad, comparan la performance de la $FastDTW$ respecto de la DTW. En series de tamaño 100, no hay mejoras significativas, para tamaño 1000, $FastDTW$ es 46 veces más rápida que DTW con $s = 0$ y del mismo orden con $s = 100$. Cuando el largo de las series es 10000, mejora 151 veces la velocidad con $s = 0$ y 7 veces con $s = 100$. Con largo 100000, 117 veces con $s = 0$ y 38 veces con $s = 100$. Concluyen que para series de largo menor que 300, no vale la pena aplicar la $FastDTW$, ya que la mejora en velocidad no es significativa y siempre resulta más precisa la DTW.

4. Clasificación con atributos series de tiempo

La clasificación supervisada es una de las áreas más importantes del aprendizaje automático. Muchos trabajos se focalizan en dominios estáticos, pero en la práctica aparecen problemas donde la variación temporal juega un rol fundamental, por ejemplo, en áreas como el reconocimiento de voz, reconocimiento de signos de lenguaje de sordos, fallos en procesos industriales, análisis de electrocardiogramas, diagnóstico de enfermedades, etc. Pasemos entonces, al problema de clasificar un conjunto de ejemplos que están caracterizados por series de tiempo.

Supongamos que tenemos un conjunto E de ejemplos de la forma

$$E = \{e_1, e_2, ..., e_n\} \text{ con } e_i = (X_i, Y_i) \text{ y } n \in \mathbb{N},$$

donde Y_i es discreta

$$Y_i \in \{H_1, ..., H_p\} \subset \mathbb{N},\ p \in \mathbb{N},$$

y cada atributo X_i consta de m de series de tiempo

$$A_1^i, A_2^i, ..., A_m^i \text{ con } m \in \mathbb{N}$$

con

$$A_j^i = a_1, ..., a_{I_j} \ \forall\ j = 1, ..., m \text{ e } I_j \in \mathbb{N}.$$

Muchas de las técnicas que se han desarrollado, comienzan con una etapa de preprocesamiento de los datos, en la cual se extraen características o patrones de las series, para luego aplicar algún método de clasificación tradicional.

Por ejemplo, en [**26**, Kadous 1999] se plantea un método denominado TClass, que combina algunas características globales de las series, como por ejemplo el promedio, máximo o mínimo global, con otras locales. Estas últimas, se obtienen mediante la extracción de eventos llamados PEPs (parametrised event primitives), que representan características temporales de los eventos por medio de parámetros. Luego,

para cada PEPs se trabaja en el correspondiente espacio del parámetro y por medio de clustering se obtienen nuevas características. Al final, se juntan estas características locales extraídas con las globales y se aplica un algorítmo de árbol de decisión C4.5 [**37**, Quinlan 1993] o un NB (Naive Bayes).

En [**18**, Geurts 2001], también se plantea un preprocesamiento de las series para la extracción de patrones. A su vez, como el método planteado genera una gran cantidad de patrones, se propone previamente obtener una representación reducida de las series, mediante un ingenioso árbol de decisión. Con los patrones obtenidos, se instrumenta un árbol de decisión, que en cada nodo contiene una tres-upla (A, p, θ), donde A es un atributo, p un patrón y θ un umbral. Mediante un test de la forma $d(p, A) < \theta$, se resuelve si un elemento e contiene o no el patrón p en el atributo A. En base a esa regla se determina la partición del nodo. En cada nodo se recorren todos los atributos, patrones y umbrales para determinar la mejor partición. El autor destaca como principal mérito del método, la interpretabilidad de las reglas resultantes.

Muchos critican, que estas etapas de preprocesamiento, dependen en gran medida de los datos específicos con los que se desea trabajar, lo que pude significar un escollo.

Otros trabajos más recientes usan el método $1NN$ (One Nearest Neighbor), midiendo la similaridad entre series con la DTW, lo llaman $1NN - DTW$. En [**52**, Xi y otros 2006] se postula, que difícilmente se alcance con otras metodologías un mejor resultado en términos de precisión. A su vez, como marcan la dificultad del tiempo de cálculo, que resulta de aplicar reiteradamente la DTW, proponen una forma de acelerar el método $1NN - DTW$. Plantean aplicar "Numerosity reduction", técnica que se basa en eliminar una gran cantidad de elementos de la base de datos de entrenamiento, sin que medie una importante pérdida de precisión. Utilizan un algoritmo que denominan $AWARD$ (Adaptative Warping Window), que combina la reducción, con un manejo adecuado de las restricciones de ventana de la DTW. Afirman que comprobaron empíricamente, a través de una gran cantidad de experimentos con variadas bases de datos, que a medida que aumenta el tamaño de la base de datos, el ancho de la ventana óptimo disminuye. Es así que $AWARD$, adapta el tamaño de la ventana que restringe la trayectoria de la DTW, aumentándolo a medida que se van descartando elementos.

Autores como J.J. Rodriguez y C.J Alonso desarrollan otras metodologías. Definen predicados, que caracterizan importantes propiedades de las series y plantean métodos para determinarlos. Dividen a estos

predicados en dos, los que se basan en el valor de una función en un intervalo y los que utilizan alguna medida de similidaridad entre series. En el primer caso, estos predicados pueden ser funciones como el promedio, máximo, mínimo, incremento, etc., de la serie en un intervalo. Estas funciones se comparan con un determinado umbral, por ejemplo, ¿el máximo en el intervalo I es mayor al umbral θ ? También se pueden determinar regiones en el dominio de las variables y comprobar si "siempre", "alguna vez", o en "cierto porcentaje" la variable pertenece a una región. Mediante un proceso detallado en [**43**, Rodriguez - Alonso, 2000], se seleccionan algunos de estos predicados. En el segundo caso, los predicados son de la forma $d(Q, R) < \theta$, donde d es una medida de similaridad (distancia euclidiana o DTW), θ un umbral, R una serie de referencia y Q la serie a comparar. Luego, mediante una técnica de Boosting, se construye un clasificador agregado que toma a los mencionados predicados como clasificadores base, ver [**42**, Rodriguez - Alonso, 2004]. Otra posibilidad descripta en este mismo artículo, es construir un árbol de decisión. Tomando las funciones de los clasificadores base, utilizados en el clasificador boosting, y dejando de lado los pesos y umbrales, consideramos una nueva base de datos. Esta consiste en los mismos elementos originales, pero ahora los atributos son los valores asignados por dichas funciones. Con esta nueva base se construye el árbol. También plantean, como alternativa a los árboles de decisión, usar SVM (Machine Vector Support), ver [**44**, Rodriguez - Alonso, 2004, 2].

CAPíTULO 5

Árboles de decisión con atributos series de tiempo

En este capítulo trataremos el tema central de nuestro trabajo, desarrollollar un método de árboles de decisión, cuando las variables de entrada se presentan en forma de series de tiempo y la variable de salida es unidimensional, discreta o continua. Nos planteamos, implementar un procedimiento que en forma directa permita construir el clasificador, sin pasar por etapas de pre-procesamiento de los datos, de manera que los resultados sean sencillos de interpretar.

Encontramos dos artículos que van en dicha dirección, [**53**, Yamada y otros, 2003] y [**42**, Rodriguez y Alonso, 2004], ambos sobre árboles de clasificación.

En el primero, los autores desarrollan y aplican su algoritmo al diagnóstico médico de la enfermedad hepatitis, aunque también reportan resultados sobre otro tipo de problemas, reconocimiento de un lenguaje por signos utilizado por la comunidad australiana de sordomudos (ASL, *Australian Sign Language*), y detección de personas alcohólicas mediante análisis de ECG. Fue desarrollado por un equipo de ingenieros en computación y médicos, donde estos últimos, estuvieron fuertemente interesados en que los resultados fueran de fácil interpretación.

En el otro artículo, se implementan algoritmos que en parte, ya fueron comentados en la Sección 4 del Capítulo 4, Página 56. Las aplicaciones están hechas sobre algunas bases de datos generadas artificialmente, un problema de reconocimiento de voz de vocales japonesas y otro de ASL.

En la siguiente Sección, explicaremos las ideas básicas del método desarrollado, que llamamos **AST** (**Á**rboles con **S**eries de **T**iempo).

1. Principios del método AST

La estructura principal del método que implementamos se basa en las ideas de CART (ver Capítulo 3), y en el algoritmo desarrollado en [**53**, Yamada y otros, 2003]. Fue implementado en el programa R [**41**, R Lenguaje V 2.9.2, 2009].

En nuestro caso, la variable de salida es unidimensional, por lo cual, todo lo que concierne a los criterios de impureza de los nodos,

lo podemos tomar en forma similar a CART. La diferencia escencial, radica en la manera de evaluar la similaridad entre las variables de entrada. Si bien, podríamos tomar las series de tiempo como vectores y aplicar directamente CART (ver Capítulo 6, Sub-sección 3.4), nuestro propósito es construir el predictor, manteniendo el carácter temporal de los datos. Por los motivos que hemos intentado explicar anteriormente, especialmente en la Sub-sección 2.1 del Capítulo 4, la medida de similaridad que nos parece más adecuada es la DTW.

1.1. Reglas de partición

. Supongamos que tenemos un conjunto E de ejemplos de la forma

$$E = \{e_1, e_2, ..., e_n\} \text{ con } e_i = (X_i, Y_i) \ y \ n \in \mathbb{N},$$

donde cada X_i está determinado por m atributos series de tiempo

$$A_1^i, A_2^i, ..., A_m^i \text{ con } m \in \mathbb{N},$$

con

$$A_k^i = a_1, ..., a_{h_k} \ \forall \ k = 1, ..., m \text{ y } h_k \in \mathbb{N}.$$

Por medio de la medida de similaridad DTW y de:

- un elemento **de referencia** e_i del nodo t que deseamos partir
- un **atributo** A_k^i de e_i
- un **umbral** $\vartheta \in \mathbb{R}$, $\vartheta > 0$

construimos particiones

$$s\left(e_i, A_k^i, \vartheta\right)$$

a partir de la siguiente regla que denominamos $\mathcal{R}\left(e_i, A_k^i, \vartheta\right)$:

Regla $\mathcal{R}\left(e_i, A_k^i, \vartheta\right)$

1. si $e_j \in t$ verifica $DTW\left(A_k^i, A_k^j\right) \leq \vartheta \Longrightarrow e_j \rightarrow t_I$
2. si $e_j \in t$ verifica$DTW\left(A_k^i, A_k^j\right) > \vartheta \Longrightarrow e_j \rightarrow t_D$

Al igual que en CART, fijamos un criterio de bondad de una partición (ver Sub-secciónes 2.2 y 3.1 del Capítulo 3) y evaluamos todas las posibles, de manera de elegir la mejor. Para eso, debemos recorrer todos los elementos del nodo, todos los atributos y todos los umbrales factibles. En resumen, en cada nodo t del árbol candidato a partirse, aplicamos el siguiente procedimiento, que denominamos $\mathcal{PI}$:

$\mathcal{PI}$

1. para $k = 1$ hasta m
2. para $i = 1$ hasta $card(t)$
3. para todo ϑ
4. obtener $s\left(e_i, A_k^i, \vartheta\right)$ con la regla $\mathcal{R}\left(e_i, A_k^i, \vartheta\right)$
5 calcular la bondad de la partición $s\left(e_i, A_k^i, \vartheta\right)$
6. elegir la mejor partición

En forma similar a lo que ocurre en CART (ver Sub-sección 2.1 del Capítulo 3), si bien los posibles umbrales son infinitos, en la práctica basta considerar los valores que efectivamente toma la $DTW\left(A_k^i, A_k^j\right)$, por lo cual, lo que efectivamente haremos es el procedimiento $\mathcal{PII}$:

$\mathcal{PII}$

1. para $k = 1$ hasta m
2. para $i = 1$ hasta $card(t)$
3. para $l = 1$ hasta $[card(t) - 1]$
4. calcular $DTW\left(A_k^i, A_k^l\right) = \vartheta_l$ y ordenar $\vartheta_1 \leq .... \leq \vartheta_{_{card(t)-1}}$
5. para $l = 1$ hasta $[card(t) - 1]$
6. obtener $s\left(e_i, A_k^i, \vartheta_l\right)$ con la regla $\mathcal{R}\left(e_i, A_k^i, \vartheta_l\right)$
7 calcular la bondad de la partición $s\left(e_i, A_k^i, \vartheta_l\right)$
8. elegir la mejor partición

1.2. Criterio de partición de un nodo

. Existen distintas maneras de medir la bondad de las particiones, de manera de elegir la que mejor contribuya al aumento de la homogeneidad, de los nodos hijos respecto al nodo padre.

Para el caso de clasificación, algunas ya fueron planteadas en la Sub-sección 2.2 del Capítulo 3. En AST optamos por la **medida de entropía** (ver Página 23), aunque hicimos pruebas con el índice de Gini (ver, Idem), con el Gain Ratio de Quinlan ([**36**, Quinlan, 1986]) y con el criterio planteado en [**31**, López de Mántaras]. Los resultados fueron similares en todos los casos, lo que reafirma lo dicho por Breiman para CART [**3**, Breiman y otros, 1984, pag 38], en cuanto a que la construción del predictor, no parece ser muy sensible a la elección de la medida de impureza.

En el caso de regresión y también como en CART, utilizamos el error cuadrático medio (ver Sub-sección 3.1 del Capítulo 3, Página **??**)

1.3. Reglas de parada del algoritmo

. Utilizamos dos criterios para detener el crecimiento del árbol, es decir para determinar cuándo dejamos de partir un nodo y lo declaramos terminal (hoja). El primero, al igual que en CART, refiere a que si la

cantidad de individuos en un nodo es muy pequeña, entonces no vale la pena partirlo. El segundo, consiste en determinar si la ganancia de impureza generada por la partición óptima, es realmente significativa, al punto que justifique partirlo.

En resumen, dados los umbrales *mindev* y *minsize*, declaramos como hoja a un nodo t si

$$card\,(t) \leq minsize \text{ ó } \Delta i\,(s^*, t) \leq mindev$$

($\Delta i\,(s^*, t)$ definido como en la Sub-sección 2.2 del Capítulo 3, Página 23)

1.4. Cálculo de la DTW

. La dinámica del método requiere el cálculo de

$$DTW\left(A_k^i, A_k^j\right) \quad \forall\ i, j = 1, ..., n, \text{ con } i < j \text{ y } \forall\ k = 1, ..., m$$

Incluso, cada una de esas distancias, puede requerirse muchas veces a lo largo del algoritmo, por lo cual lo que hacemos, es calcular previamente a la contrucción del árbol una matriz tridemensional D, de tamaño $n \times n \times m$, donde

$$D\,(i, j, k) = DTW\left(A_k^i, A_k^j\right).$$

Como se verifica

$$DTW\left(A_k^i, A_k^j\right) = DTW\left(A_k^j, A_k^i\right) \quad \forall\ i, j = 1, ..., n \text{ y } \forall\ k = 1, ..., m$$

y

$$DTW\left(A_k^i, A_k^i\right) = 0 \quad \forall\ i = 1, ..., n \text{ y } \forall\ k = 1, ..., m,$$

la matriz D es simétrica y todos los elementos de su diagonal son cero, por lo cual la cantidad de veces que efectivamente se requiere calcular la DTW es

$$\frac{n\,(n-1)}{2} \times m.$$

Una vez obtenida la matriz D, el algoritmo que construye el árbol recurre a ella, todas las veces que sea necesario.

2. Reducción en el número de particiones utilizadas

La parte medular del algoritmo y la que consume mayor tiempo de ejecución, es el procedimiento $\mathcal{PII}$. Tenemos que correr en cada nodo t, $(m \times [card\,(t)] \times [card(t) - 1])$ veces los renglones 6 y 7, que calculan las particiones y miden su bondad. La observación empírica, de que en cada nodo, varios elementos de referencia generan particiones con similar bondad, nos condujo a implementar un algoritmo, que solamente utiliza como individuos de referencia, una parte del total de individuos del nodo. Es decir, en lugar de recorrer como en el renglón 2 de $\mathcal{PII}$

todos los individuos del nodo, elegimos al azar una proporción *prop* (con $0 < prop < 1$) y aplicamos el procedimiento $\mathcal{PIII}$:

$\mathcal{PIII}$

1. elegimos una proporción de individuos a utilizar, *prop*
2. para $k = 1$ hasta m.
3. para $i \in sort = prop \times card(t)$
4. para $l = 1$ hasta $[card(t) - 1]$
5. calcular $DTW\left(A_k^i, A_k^l\right) = \vartheta_l$ y ordenar $\vartheta_1 \leq .... \leq \vartheta_{card(t)-1}$
6. para $l = 1$ hasta $[card(t) - 1]$
7. obtener $s\left(e_i, A_k^i, \vartheta_l\right)$ con la regla $\mathcal{R}\left(e_i, A_k^i, \vartheta_l\right)$
8. calcular la bondad de la partición $s\left(e_i, A_k^i, \vartheta_l\right)$
9. elegir la mejor partición

Aplicando el algoritmo implementado en $\mathcal{PIII}$, efectivamente reducimos el tiempo de ejecución en cada nodo del árbol, en una proporción similar a *prop*, ya que ahora corremos los renglones 7 y 8 de $\mathcal{PIII}$, $(m \times prop \times [card\,(t)] \times [card(t) - 1])$ veces. Si bien, se podría pensar que esto va en desmedro de la precisión del clasificador, algunos ejemplos indican otra cosa. Como mostramos en algunas aplicaciones de la Sección 3 del Capítulo 6, si medimos la performance con el estimador por muestra test o validación cruzada, obtenemos, que para valores de *prop* entre 0,05 y 0,2 esta mejora Es a partir de 0,01 que la performance empeora. No ocurre lo mismo si hacemos la evaluación con el estimador por resustitución, en este caso, desmejora uniformemente a medida que disminuye *prop* (ver tabla en Figura 2 del Capítulo 6).

Pensamos que este comportamiento, se explica por el ya mencionado problema del sobre-ajuste. Al recorrer todos los individuos de referencia, en cada nodo encontramos la partición óptima y por medio de éstas, el árbol clasificador, pero al tomar nuevos datos, en forma independiente de la muestra de entrenamiento con la cual se obtuvo el clasificador, éste no funciona tan bien, ya que fue construido de forma muy "ajustada" a la muestra de entrenamiento. En cambio, tomando solamente una parte de los individuos del nodo como referencia, las sucesivas particiones óptimas en cada nodo, permiten obtener un árbol clasificador menos ajustado a la muestra de entrenamiento, que parece recoger mejor la naturaleza del fenómeno.

3. Agregación de modelos. Bagging.

Debido a la naturaleza inestable, que en general presentan los métodos basados en CART, decidimos implementar un modelo agregado,

utilizando a los predictores AST como clasificadores base (ver Subsección 6.2 del Capítulo 2). Optamos por aplicar un procedimiento de Bagging, que denominamos AST-BAG.

En primer lugar, partimos al azar la muestra inicial de los datos en dos subconjuntos, la muestra que usaremos para entrenamiento, M^e y la muestra M^p, que permanecerá fija como muestra de prueba. Luego, determinamos muestras boostrap M^e_B, con $1 \leq B \leq K$, sorteando al azar y con reposición, dentro del total de los elementos de M^e. Con cada una de las K muestras, entrenamos un predictor AST. Al final, computamos el error bagging por muestra de prueba R^{mp}, clasificando cada elemento de la muestra M^p, mediante el voto mayoritario entre las K predicciones, obtenidas de los correspondientes K árboles clasificadores AST. Analogamente obtenemos el error bagging de resustitución R^{res}, clasificando los elementos de la muestra M^e.

CAPíTULO 6

Experimentación

Nos abocamos a la búsqueda de bases de datos de referencia, que nos permitieran experimentar nuestro método y compararnos con otras metodologías, tanto en clasificación como en regresión.

Encontramos, que para el problema de clasificación con atributos series de tiempo, no existe consenso, en la comunidad de autores que estudian este tipo de problemas, acerca de una base de referencia. Solamente constatamos, que ciertas bases de datos se utilizan reiteradamente y varias publicaciones reportan resultados sobre ellas. Dentro de este grupo, se encuentra la denominada **base CBF**, con la cual decidimos trabajar.

En primer lugar, corrimos el programa AST, para los distintos valores de los parámetros del modelo. Comparamos nuestros resultados con los publicados por otros autores. También comparamos AST con el tradicional CART. Por último, implementamos un método bagging, agregando clasificadores base AST.

Para el caso de regresión con atributos series de tiempo, apelamos a una base de datos, de tráfico de correo electrónico en una red de internet. Tratamos de predecir una variable que mide ese tráfico, en función del comportamiento anterior de esa variable, representado por una serie de tiempo.

1. Base CBF

Esta base de datos artificiales fue introducida por [**45**, Saito, 1994] y usada entre otros por , [**26**, Kadous, 1999], [**43**, Rodríguez y Alonso, 2000], [**18**, Geurts, 2001], [**8**, Chu y otros, 2002], [**42**, Rodríguez y Alonso, 2004], [**19**, Geurts y Wehenkel, 2005], [**27**, Kadous y Sammut, 2005]. Algunos autores, como [**8**, Chu y otros, 2002], llegan a afirmar que *"The most commonly studied benchmark dataset is Cylinder-Bell-Funnel"*.

La base de datos

$$E = \{e_1, e_2, ..., e_n\} \text{ con } e_i = (X_i, Y_i) \text{ y } n \in \mathbb{N},$$

está formada por 3 clases de elementos, **cilindro** (*cylinder*), **campana** (*bell*) y **embudo** (*funnel*), que en adelante las denominamos **C, B y F** respectivamente. Tenemos pues que

$$Y_i \in \{\mathbf{C}, \mathbf{B}, \mathbf{F}\}$$

y X_i, está caracterizada por un atributo del tipo serie de tiempo, definido de la siguiente manera:

$$\begin{aligned} A^i(t) &= (6+\mu)\,\mathbf{1}_{[a,b]}(t) + \epsilon(t) && \text{si } Y_i = \mathbf{C} \\ A^i(t) &= (6+\mu)\,\mathbf{1}_{[a,b]}(t)\,\frac{(t-a)}{(b-a)} + \epsilon(t) && \text{si } Y_i = \mathbf{B} \\ A^i(t) &= (6+\mu)\,\mathbf{1}_{[a,b]}(t)\,\frac{(b-t)}{(b-a)} + \epsilon(t) && \text{si } Y_i = \mathbf{F} \end{aligned}$$

donde:

- $t \in [1, 128]$
- $\mathbf{1}_{[a,b]}$ representa la función indicatriz en el intervalo $[a, b]$
- $\epsilon(t)\ \forall\ t = 1, ..., 128$ y μ tienen distribución $N(0, 1)$
- a tiene distribución uniforme discreta $U[16, 32]$
- $(b - a)$ tiene distribución uniforme discreta $U[32, 96]$

Como se aprecia en la Figura 1, las tres clases tienen un comportamiento marcadamente distinto entre a y b; la clase **C** presenta un patrón del tipo meseta, la clase **B** un incremento gradual y la clase **F** un descenso gradual. Por otra parte, con esta base, se intenta simular algunos comportamientos típicos en dominios temporales, a través de la variación en amplitud (que resulta del efecto de μ), del ruido producido por $\epsilon(t)$ y de la significativa variación temporal en el inicio y final del patrón típico de cada clase, ya que a y $(b - a)$ se eligen sorteando distribuciones uniformes de longitud considerable. Un buen clasificador, debería captar los patrones característicos de cada clase, a pesar del ruido y de las importantes deformaciones en el eje del tiempo.

Al igual que en todos los trabajos referidos que usan esta base, tomamos 798 individuos, 266 de cada clase, le llamamos **base CBF-798**. Para la evaluación del error de clasificación, utilizamos los estimadores por muestra de prueba, (532 para entrenamiento y 266 para muestra de prueba) y de validación cruzada 10.

2. Implementación

Todos los algoritmos fueron implementados mediante el software **R** [**41**, R Lenguaje V 2.9.1, 2009]. Como explicamos en el Capítulo anterior, tenemos dos etapas bien diferenciadas. En la primera, mediante el paquete **"dtw"**[**22**, Giorgino, 2009] determinamos una matriz que

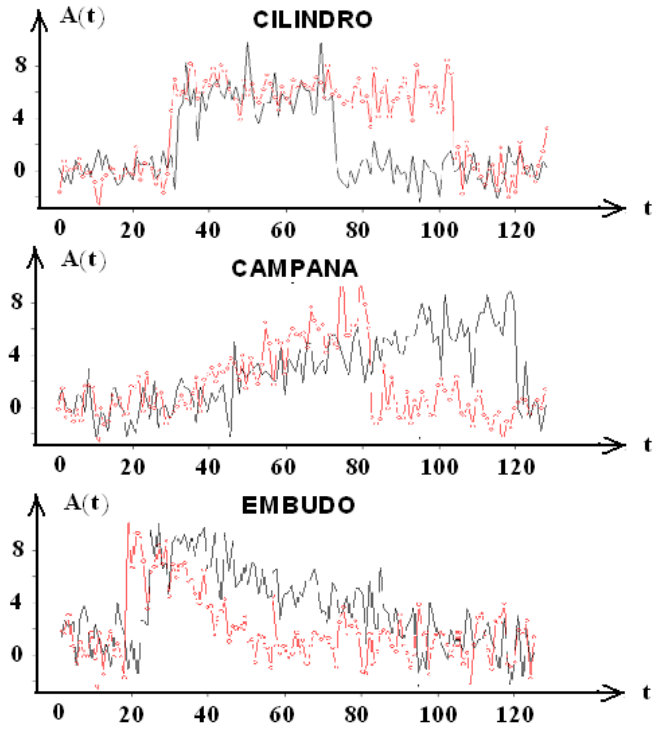

FIGURA 1. *Dos ejemplos de cada clase de la base CBF.*

contiene todas las medidas DTW entre series, necesarias para la construcción y evaluación del árbol. Este paquete, permite variar el algoritmo de cálculo de la DTW, que como se planteó en la Sección 2 del Capítulo 4, admite diversas formas, según la elección de las restricciones de ventana y pendiente, de los coeficientes de ponderación y de la normalización. Luego de algunas pruebas, optamos por la opción "symmetric1"del referido paquete, que corresponde a la fórmula 2.13 (Página 50), sin restricciones de ventana ni de pendiente y sin normalizar.

La segunda etapa, consiste en correr los programas(*) elaborados para construir y evaluar el árbol, según los criterios analizados en la Sección 1 del Capítulo 5.

Todos los resultados que exponemos para la base CBF-798 toman como criterios de parada $minsize = 5$ y $mindev = 0{,}01$ (ver Subsección 1.3 del Capítulo 5).

(*) Las primeras versiónes de estos programas fueron elaboradas conjuntamente con el profesor Juan Piccini (Instituto de Matemática de la Facultad de Ingenieria de la Universidad de la República, Montevideo-Uruguay), en el marco

de un curso dictado en el año 2007 por el profesor Badih Ghattas (Instituto de Matemática de Luminy, Marsella-Francia)

3. Resultados del modelo AST

Repetimos 100 veces cada experimento, volviendo a sortear las muestras de entrenamiento y prueba. Calculamos los promedios, de los estimadores del error de clasificación y sus desviaciones estándar. También promediamos la cantidad de hojas de los árboles clasificadores. Presentamos los resultados, en la tabla de la Figura 2, para distintos valores de la proporción de elementos (*prop*), que se utilizan como individuoas de referencia en cada nodo. Como señalamos en la Sub-sección 2 del Capítulo 5, a medida que *prop* disminuye, los estimadores honestos del error de clasificación, R^{mp} y R^{vc}, mejoran para valores de *prop* entre 0,2 y 0,05, y empeoran a partir de 0,01. También destacamos, que para valores de *prop* de 1 hasta 0,05, el tamaño del árbol es el menor posible (3 hojas para un problema de 3 clases).

prop	R^{mp}	hojas	R^{vc}	hojas	R^{res}	hojas
1	1.04 ± 0.41	3	1.2 ± 0.9	3	0	3
0.5	0.92 ± 0.48	3	1.1 ± 1.05	3	0	3
0.2	$\mathbf{0.81 \pm 0.56}$	3	$\mathbf{0.75 \pm 1.12}$	3	0	3
0.1	0.81 ± 0.6	3	0.77 ± 1.2	3	0	3
0.05	0.89 ± 0.66	3	0.82 ± 1.2	3	0.04 ± 0.08	3
0.02	0.99 ± 0.7	3.06	1.08 ± 1.32	3.14	0.19 ± 0.18	3.02
0.01	1.43 ± 0.9	3.61	1.22 ± 1.46	3.61	0.37 ± 0.29	3.4

FIGURA 2. *Tabla conteniendo los errores por muestra de prueba, validación cruzada y resustitución, para distintos valores de prop. En cada caso, se promediaron 100 corridas del método AST, para una base CBF-798.*

3.1. Ejemplo del método AST con prop=0.1

. Un ejemplo del algoritmo AST, para la base CBF-798 con $prop = 0{,}1$, se muestra en la Figura 3.

En este caso, el árbol construido, clasifica sin error los 532 individuos de la muestra de entrenamiento ($R^{res} = 0$). Con los restantes 266 individuos de la muestra test, el error es $R^{mp} = 0{,}38\,\%$, es decir que clasifica mal 1 elemento.

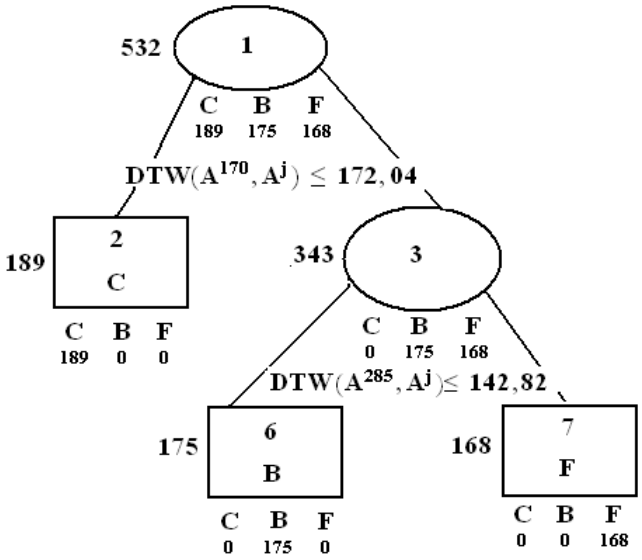

FIGURA 3. *Árbol construido por el método AST, con una base de datos CBF-798, 532 individuos para la muestra de entenamiento, 266 para la muestra test y prop = 0.1. A la izquierda de cada nodo t, se indica la cantidad total de individuos, debajo la cantidad de cada clase, en el interior el número del nodo y si es hoja su clase. Para los nodos no terminales, se indica la regla de partición. Los errores son $R^{res} = 0$ y $R^{mp} = 0.38\,\%$ (1 elemento mal clasificado en 266). Individuo de referencia en el nodo 1, con $Y_{170} = C$, y en el nodo 3, con $Y_{285} = B$.*

3.2. Ejemplo del método AST con prop=0.01

. Otro ejemplo del algoritmo AST, para la base CBF-798, pero construido con *prop* = 0,01, se muestra en la Figura 4. Respecto al caso anterior la performance es peor, el error sobre la muestra de entrenamiento es 0,94 % (5 individuos mal clasificados en 532) y sobre la muestra de prueba 2,63 % (7 individuos mal clasificados en 266). Esta merma en la performance parece razonable, ya que *prop* = 0,01 en este caso, significa tomar como individuos de referencia, para elegir la mejor partición, en el nodo 1 solamente 5 elementos, en el nodo 3 solo 3 y en el 6 apenas 1.

Por otra parte, observamos que es discutible la conveniencia de partir el nodo 6, ya que al hacerlo aumentamos la complejidad del clasificador y apenas mejoramos el error de resustitución (de 5 mal clasificados a 4). Si la cantidad mínima de individuos por hoja (*minsize*)

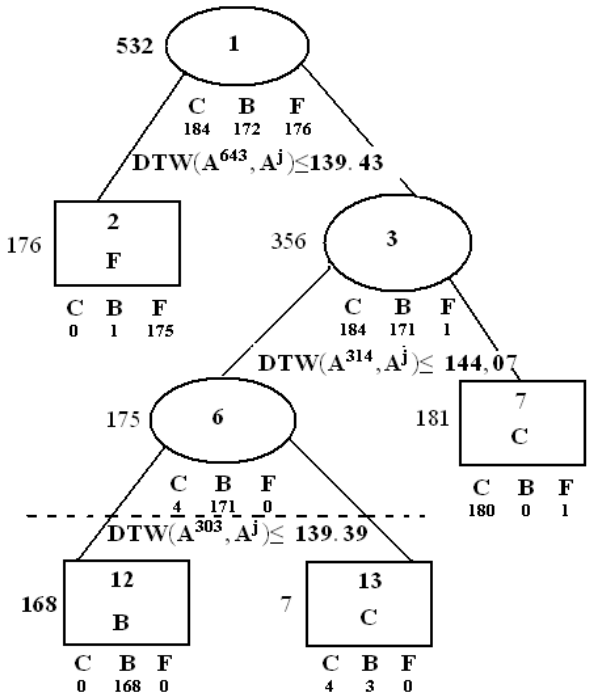

FIGURA 4. *Árbol construido por el método AST, con una base de datos CBF-798, 532 individuos para la muestra de entenamiento, 266 para la muestra test y prop=0.01. Se obtuvieron errores* $R^{res} = 0.94\,\%$ *(5 elementos mal clasificados en 532) y* $R^{mp} = 2.63\,\%$ *(7 elementos mal clasificados en 266). Individuo de referencia en el nodo 1, con* $Y_{643} = F$*, en el 3, con* $Y_{314} = B$ *y en el 6, con* $Y_{303} = B$*. La línea horizontal punteada, indica donde podría detenerse el proceso de partición, de forma de obtener un árbol más sencillo, pero sin pérdidas significativas en su performance.*

fuera mayor a 7 (en lugar de 5), entonces el nodo 6 no se partiría. Un efecto similar, podría lograrse mediante un procedimiento de poda como en CART (ver Sub-sección 2.7 del Capítulo 3), o exigiendo como criterio para aceptar partir un nodo, un umbral (*mindev*) menor para la ganancia de impureza (ver reglas de parada en la Sub-sección 1.3 del Capítulo 5).

3.3. Comparación con otros métodos

. En el siguiente cuadro, presentamos resultados publicados por otros

autores, para la base de datos CBF-798. Algunos de estos métodos ya fueron comentados en el Capítulo 4.

Publicación y método. Base CBF-798	$\mathbf{R}^{vc}$
[**26**, Kadous, 1999]. TClass-C4.5	$1{,}9 \pm 0{,}57$
[**26**, Kadous, 1999]. TClass-NB	$3{,}67 \pm 0{,}61$
[**18**, Geurts, 2001]. Tree	$1{,}17 \pm 1{,}67$
[**18**, Geurts, 2001]. 1-NN	$0{,}33 \pm 0{,}67$
[**42**, Rodríguez y Alonso, 2004]. Tree-DTW	$1{,}62 \pm 1{,}65$
[**42**, Rodríguez y Alonso, 2004]. Tree-intervalos	$2{,}27 \pm 1{,}8$
[**42**, Rodríguez y Alonso, 2004]. Boosting-DTW	$0{,}38 \pm 0{,}61$
[**42**, Rodríguez y Alonso, 2004]. Boosting-intervalos	$1{,}13 \pm 1{,}23$
[**42**, Rodríguez y Alonso, 2004]. Tree-intervalos y DTW	$0{,}87 \pm 1{,}02$
[**52**, Xi y otros, 2006]. 1-NN-DTW	0
[**27**, Kadous y Sammut, 2005]. TClass with AdaBoost/J48	0
[**27**, Kadous y Sammut, 2005]. Naive segmentation	0
[**27**, Kadous y Sammut, 2005]. Hidden markov model	0

3.4. Comparación con CART

. A los efectos de destacar las ventajas de las metodologías específicas para dominios temporales, analizamos el desempeño de CART sobre la misma base de datos CBF-798 a la que le aplicamos AST, tomando como variables explicativas, los 128 valores de la serie de tiempo A. Es así que dada la base de datos

$$E = \{e_1, e_2, ..., e_{798}\} \text{ con } e_i = (X_i, Y_i) \text{ , y } Y_i \in \{C, B, F\}$$

ahora

$$X_i \in \mathbb{R}^{128} \text{ con } X_i = \left[A^i(1), ..., A^i(128)\right].$$

Utilizando el paquete "**tree**" [**40**, Ripley, 1996] del programa R, obtuvimos que los estimadores de los errores de clasificación por muestra de prueba y validación cruzada son $7{,}55 \pm 2{,}7\,\%$ y $7{,}98 \pm 1{,}54\,\%$ respectivamente y la cantidad de hojas 11, promediando como antes, 100 corridas.

Si consideramos en particular, las mismas muestras de entrenamiento y prueba que en el ejemplo de la Sub-sección 3.2, obtenemos $R^{mp} = 9{,}4\,\%$. Para interpretar mejor la estructura del árbol que construye CART, hacemos una poda por mínimo-costo-complejidad (ver Sub-sección 2.7 del Capítulo 3), utilizando la función "**prune.tree**" para elegir un árbol de 3 hojas y simplificar el análisis. El resultado se muestra en la Figura 5. Los errores son bastante altos, $R^{res} = 10{,}9\,\%$ y $R^{mp} = 11{,}65\,\%$, intentemos analizar porqué. Observemos que el nodo

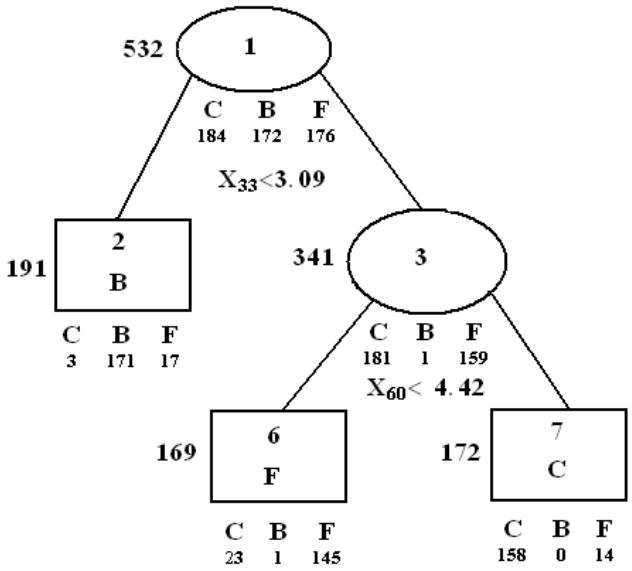

FIGURA 5. *Arbol construído por el método CART con una base de datos CBF* $-\, 798, 532$ *individuos para la muestra de entenamiento y 266 para la muestra test. Se aplicó una poda por costo-mínimo-complejidad y se eligió el árbol de 3 hojas. Los errores son* $R^{res} = 10.9$ *% (58 elementos mal clasificados en 532) y* $R^{mp} = 11.65$ *% (31 elementos mal clasificados en 266).*

1, se parte en función de lo que ocurre en $t = 33$. Si alteraramos la definición de los atributos series de tiempo de la base CBF, quitando los elementos aleatorios y fijando:

- $a = E\left(U\left[16, 32\right]\right) = 24$
- $b = E\left(U\left[16, 32\right]\right) + E\left(U\left[32, 96\right]\right) = 88$
- $\mu = \epsilon = E\left(N\left[0, 1\right]\right) = 0$

entonces efectivamente los individuos de clase **C** y **F** siempre tomarían en $t = 33$ valores mayores a 3,09 y los de clase **B** valores menores a 3,09 (ver Figura 6). Con la "verdadera" base CBF, lo anterior ocurre con alta frecuencia, pero como se aprecia en la Figura 5, hay 3 elementos de los 184 de la clase **C** y 17 de los 176 de clase **F,** que toman valores menores a 3,09 y 1 de los 172 de clase B que toma un valor mayor que 3,09. Algo similar ocurre al partir el nodo 3, sin aleatoriedad en la base CBF, se esperaría que en $t = 60$ los individuos de clase **C,** estuvieran por encima de 4,42 y los de clase **F,** por debajo de ese valor (ver Figura 6). Sin embargo hay 23 de los 181 de clase **C** y 14 de los 159 de clase **F** que cumplen lo contrario.

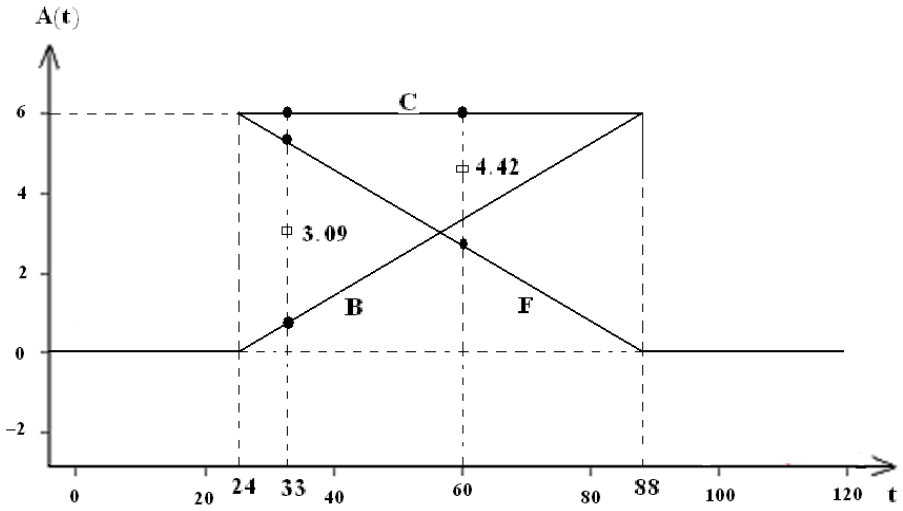

FIGURA 6. *Se representa un individuo por cada clase de una base CBF modificada, en la cual se han suprimido los elementos aleatorios, fijando* $a = 24$, $b - a = 64$ *y* $\mu = \epsilon(t) = 0 \; \forall \; t = 1, ..., 128$. *En* $t = 33$ *y* $t = 60$ *se indican los valores de los umbrales 3.09 y 4.42, correspondientes a las reglas de partición del CART considerado (ver Figura 5).*

4. Resultados del modelo AST-Bagging

En la Figura 7, presentamos en forma gráfica, los resultados promedios de los estimadores de los errores bagging de clasificación (ver Sección 3 del Capítulo 5), sobre 100 corridas de bagging, para una base CBF-798, con $prop = 0{,}01$. El eje horizontal, corresponde a la cantidad B de pasos iterativos del Bagging y en el eje vertical computamos los errores sobre las muestras de entrenamiento y de prueba (2/3 y 1/3 del total respectivamente). El error de resustitución, R^{res}, disminuye rápidamente, cae debajo de 0,01 % en el paso 9 y vale 0 a partir del paso 20 de bagging. El error por muestra de prueba, R^{mp}, cae debajo de 0,1 % a partir de la iteración 9, de 0,01 % luego del paso 21 y permanece debajo de 0,004 % a partir del paso 23.

Obtenemos entonces un modelo, que si bien no es tan claro de interpretar y resulta más costoso computacionalmente, tiene la ventaja de disminuir significativamente el error de clasificación.

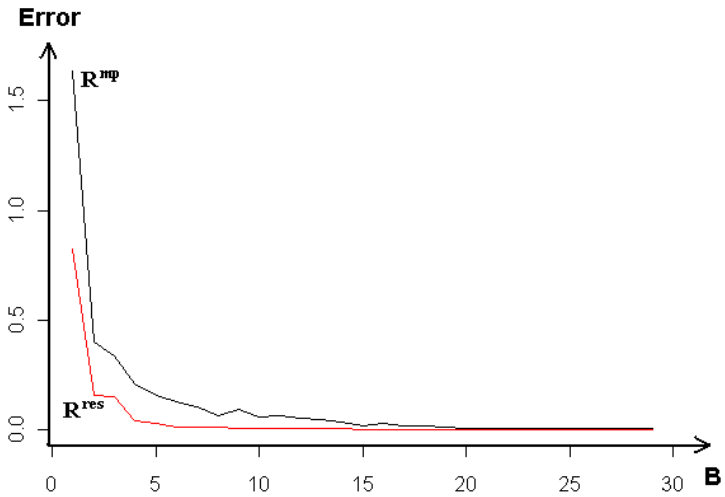

FIGURA 7. *Gráfico de los errores de resustitución y muestra de prueba, para 30 iteraciones del método AST-BAG. Se tomó un tercio de la base CBF-798 como muestra de prueba y prop=0.01. Los valores corresponden al promedio de 100 corridas.*

5. AST en regresión - Tráfico en redes de internet

La base de datos que utilizamos, se compone de una traza de valores de tráfico en una red de computación, medidos en el POP (***P**oint **O**f **P**resence*) de un ISP (***I**nternet **S**ervice **P**rovider*) de origen italiano. Estos datos, muestran una importante heterogeneidad, ya que pueden incluir archivos de voz, video, etc., que reciben usuarios finales, ya sean residenciales o de grandes empresas. Fueron tomados durante el horario diurno, el 15 de Mayo de 2007, como tráfico de "bajada", en un período de aproximadamente 2 horas 46 minutos (10000 segundos). En suma, tenemos un vector de 10000 valores medidos en Mbps (Megabits por segundos), por más detalles ver [**2**, Bermolen y Rossi, 2008].

5.1. Base de datos

. Supongamos que λ_t es la carga de tráfico en el instante t, y que tenemos el conjunto

$$\{\lambda_t : t = 1, ..., 10000\},$$

a partir del cual formamos la siguiente base de datos:

$$E = \{e_1, e_2, ..., e_{10000-d}\} \text{ con } e_i = (X_i, Y_i) \text{ y } d \in \mathbb{N},$$

donde

$$Y_i = \lambda_{i+d}$$

y

$$X_i = (\lambda_i, ..., \lambda_{i+d-2}, \lambda_{i+d-1})$$

es una serie de tiempo de largo d.

Observemos que d es un parámetro importante, ya que representa cuántos valores del pasado estamos tomando en cuenta, para predecir el tráfico en un instante t.

5.2. Resultados AST

. La implementación en general, es similar a la realizada en clasificación para la base CBF-798.

Presentamos, en la Figura 8, los valores del estimador por muestra de prueba de la raíz del error cuadrático medio (RECM), para distintos valores del parámetro d y para distintas proporciones de la parte de la muestra que usamos para entrenar al predictor ($q = card(M_{ent})/10000$). En todos los casos, $minsize = 20$, $mindev = 0{,}01$, $prop = 1$. También, para cada elección de q y d, damos el correspondiente RECM que resulta de aplicar el modelo CART, usando el paquete "tree" del programa R ([**40**, Ripley, 1996]). Aquí, tomamos el árbol predictor, resultante de la optimización mediante la poda por mínimo costo-complejidad (ver Subsecciónes 2.7 y 3.2 del Capítulo 3), aplicando la funcion "prune.tree". En cada caso se promediaron 10 corridas.

Los resultados del método AST, en este caso, no presentan grandes variaciones al cambiar *prop* (menores al 5 %). A su vez, son similares a los de CART, aunque en nuestro caso no implementamos un procedimiento de poda.

Por otro lado, quisimos comparar estos resultados con otros métodos. En [**2**, Bermolen y Rossi, 2008], se implementa un algoritmo SVM (ver Sub-sección 6.3 del Capítulo 2), a los mismos datos que utilizamos nosotros, siguiendo un riguroso estudio de los parámetros del modelo. También aplican otros modelos, como Promedios Móviles (**M***oving-**A**verage*, MA), Auto-Regresivos (**A***uto-**R**egressive*, AR) y Cambios de Nivel y Outliers (**L***evel-**S**hift and **O**utliers*, LSO). Según lo publicado, las performance de SVR y AR resultan mejores que AST y CART (del orden del 5 %) y las de MA y LSO peores.

Pensamos que la estructura de las series de tiempo tratadas, no permiten explotar las ventajas de la medida de similaridad DTW, en

		AST		CART	
q	**d**	**RMSE**	**hojas**	**RMSE**	**hojas**
0.1	**5**	**6.36**	**6**	**6.36**	**7**
0.2	**5**	**6.26**	**9**	**6.27**	**8**
0.2	**10**	**6.26**	**8**	**6.21**	**9**
0.2	**20**	**6.35**	**10**	**6.36**	**7**
0.2	**40**	**6.64**	**10**	**6.34**	**8**
0.25	**10**	**6.24**	**8**	**6.34**	**9**
0.25	**5**	**6.18**	**12**	**6.23**	**7**

FIGURA 8. *Tabla conteniendo el RMSE y la cantidad de hojas de los métodos AST y CART, para distintos valores de los parámetros q y d. En todos los casos se promediaron 10 corridas de la base "Tráfico en redes".*

lo que respecta a detectar e ignorar ciertas deformaciónes en el eje del tiempo.

En las Figuras 9 y 10, mostramos las salidas gráficas de AST y CART, correspondientes a un mismo ejemplo, donde $q = 0{,}25$ y $d = 5$. En el caso de AST, lo primero que se lee en la parte superior de la Figura 9

$$d\,(X1, 1352) < 73{,}1,$$

significa que el primer nodo se divide según la regla

$$DTW\,(X1, X_{1352}) < 73{,}1,$$

donde, recordando que representamos a la traza de tráfico como

$$\{\lambda_t : t = 1, ..., 10000\} :$$

- **X1** es la serie-atributo

  $$X_i = (\lambda_i, \lambda_{i+1}, \lambda_{i+2}, \lambda_{i+3}, \lambda_{i+4})$$

  de un individuo e_i, genérico del nodo.
- $$X_{1352} = (\lambda_{1352}, \lambda_{1353}, \lambda_{1354}, \lambda_{1355}, \lambda_{1356})$$

  es la serie-atributo del individuo e_{1352}.

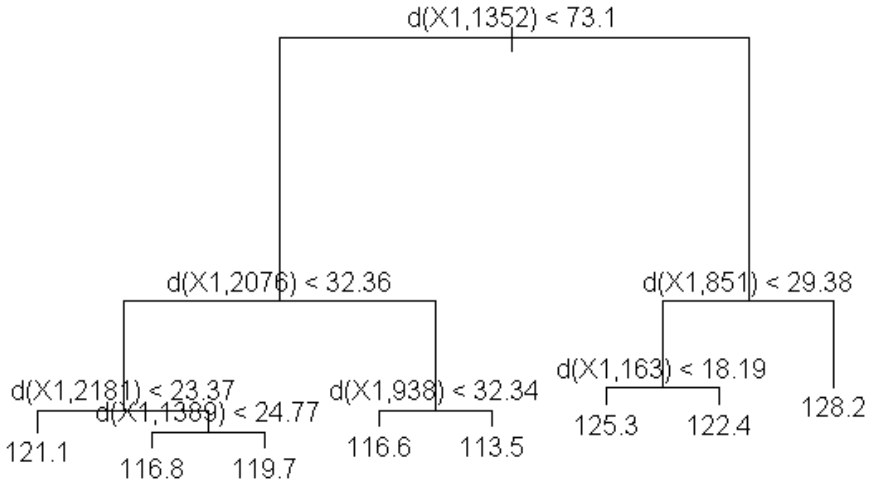

FIGURA 9. *Salida gráfica del algoritmo AST programado en R. Árbol para datos "Tráfico en redes", n=10000, q=0.25, d=5, prop=0.01. RMSE=6.28, hojas=8.*

Para el caso de CART, en la Figura 10, **X2, X3, X4, X5 y X6** representan los 5 atributos escalares λ_i, λ_{i+1}, λ_{i+2}, λ_{i+3} y λ_{i+4} del individuo e_i.

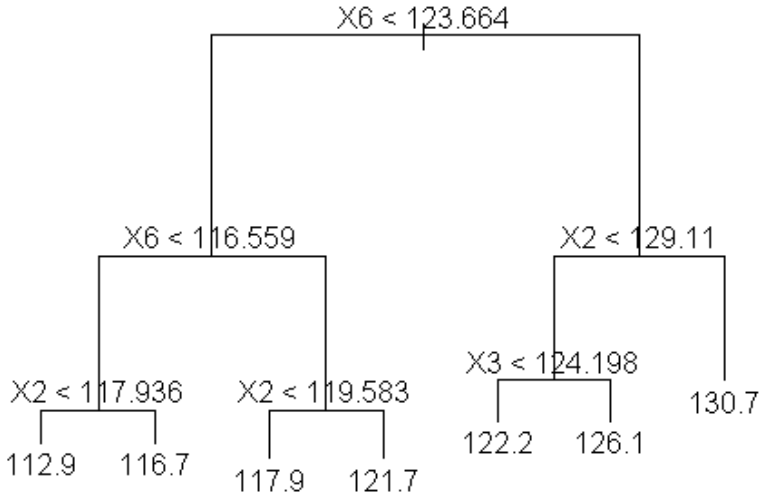

FIGURA 10. *Salida gráfica de la función "prune.tree" del paquete "tree" (programa R). Árbol para datos "Tráfico en redes", n=10000, q=0.25, d=5. RMSE=6.23, hojas=7.*

5.3. Resultados AST-Bagging

. En la Figura 11, presentamos en forma gráfica los valores de RECM para el método bagging, a partir de los clasificadores base AST y CART. Lo implementamos en forma similar a clasificación (ver Sección 3 del Capítulo 5). Para regresión, en lugar de voto mayoritario, asignamos a cada elemento de la muestra de prueba, el promedio de las predicciones obtenidas a partir de los K árboles predictores. Tomamos $d = 5$, $q = 0{,}25$, $minsize = 20$, $mindev = 0{,}05$ y $prop = 0{,}01$. Luego de algunos pasos del bagging, el RECM baja a niveles muy próximos a 6 (en el paso 7, RECM-CART=6.057 y RECM-AST=6.025), manteniendose un poco más bajo en el caso de CART (recordemos que en los resultados de la figura 8, $prop = 1$)

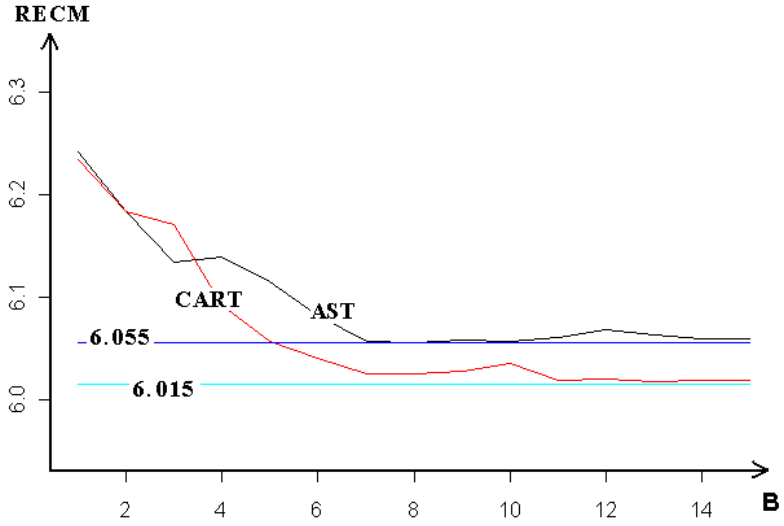

FIGURA 11. *Gráfico de los RECM para 15 iteraciones bagging de AST y CART. Base de datos "Tráfico en redes", prop=0.01, q=0.25 y d=5.*

CAPíTULO 7

Conclusiones y perspectivas

Hemos presentado un método de árboles de clasificación y regresión, para datos cuyos atributos tienen la forma de series de tiempo. Nos basamos en la estructura básica de CART, modificando las reglas de partición de los nodos.

Las series que caracterizan a los individuos son tratadas sin ningún tipo de procesamiento y las reglas de partición se refieren directamente a ellas, por lo cual la interpretación del modelo final es sencilla.

La utilización de la medida DTW, para comparar series de tiempo, es reconocida ampliamente por su eficacia y permite tener en cuenta deformaciones temporales en el eje del tiempo. También es sabido su alto costo computacional, por lo que hicimos una búsqueda en la literatura, sobre procedimientos que permitieran acelerar su cálculo.

Al igual que en CART, el método en principio es exhaustivo en la búsqueda de la mejor partición, aunque en este caso las reglas requieren además de recorrer todos los posibles umbrales y atributos, hacerlo con todos los individuos. Implementamos una manera de sortear una proporción de individuos en cada nodo. En el caso de la base CBF, el resultado es que además de acelerar los cálculos, los errores de clasificación estimados por muestra de prueba y validación cruzada son más bajos, para proporciones entre 0,05 y 0,2. Pensamos que esto se debe, a que la busqueda exhautiva genera sobreaprendizaje.

Trabajamos sobre la base CBF, ya que existen muchos trabajos con resultados publicados, en los cuales se aplican diversas técnicas, como K-NN, SVM, Boosting, etc., combinados con métodos desarrollados específicamente para este tipo de problemas. Nuestro método arroja resultados de error del orden del 1 %, cuando usamos los estimadores por muestra test y validación cruzada, pero la variabilidad es muy alta (también del orden del 1 % medido por la desviación estándar entre 100 corridas del mismo experimento). Sin embargo, aplicando una sencilla técnica de Bagging, se puede llegar a errores por muestra de prueba próximos a cero (debajo de 0,1 % en 9 pasos, de 0,01 % en 20 y de 0,004 % en 23).

En la aplicación del método AST en regresión, al problema de tráfico en redes, los resultados no muestran evidencia de que se supere la performance de otros procedimientos clásicos, como CART, SVM o AR.

Nos queda planteado, trabajar con otras bases de datos frecuentemente utilizadas en la literatura, ya que la experiencia indica que algunos métodos, pueden recoger mejor las características particulares de ciertos tipos de datos. También estamos convencidos de que nuestros programas pueden ser optimizados y que sería deseable implementar un procedimiento de poda.

También nos interesaría, adaptar el método AST, de manera de tratar las series de tiempo como datos funcionales. En ese caso, podríamos tomar criterios análogos para las reglas de partición de los nodos, que en lugar de utilizar la medida de similaridad DTW, utilicen una distancia en el correspondiente espacio de funciones. Con esta metodología, se podría intentar generalizar, alguno de los resultados teóricos conocidos sobre la consistencia de los estimadores, como los enunciados en la Sub-sección 2.10 del Capítulo 3.

Bibliografía

[1] Berndt, D., Clifford, J. (1994). *Using dynamic time warping to find patterns in time series.* AAAI-94 Workshop on Knowldge Discovery in Databases.

[2] Bermolen, P., Rossi, D. (2008): *Support vector regression for link load prediction*. Computer Networks 53(2), pp 191-201.

[3] Breiman, L., Friedman, J.,Olshen, R., and Stone, C. (1984). *Classification and Regression Trees.* Belmont, CA: Chapman & Hall.

[4] Breiman, L. (1994). *Bagging Predictors*. Machine Learning 24, pp 123-140.

[5] Breiman, L. (1996). *Stacked Regressions,* Machine Learning Vol. 24, pp 49-64.

[6] Breiman, L. (2001). *Random Forests*. Machine Learning 45, pp 5-32.

[7] Cardot, H., Ferraty, F., Sarda, P. (1999). *Functional Linear Model*. Statistics and Probability Letters 45, pp 11–22.

[8] Chu, S., Keogh, E., Hart, D., Pazzani, M. (2002). *Iterative Deepening Dynamic Time Warping for Time Series*. In Procceding of de Second SIAM Intl. Conf. on Data Mining. Arlington, Virginia, 2002.

[9] Shawe, J., Cristianini, N. (2004). *Kernel Methods for Pattern Analysis*. Cambridge University Press.

[10] Cuevas, A., Febrero, M., Fraiman, R. (2004). *An ANOVA Test for Functional Data*. Computational Statistics and Data Analysis 47, pp 111–122.

[11] Deville, J. (1974), *Méthodes statistiques et numeriques de l'analyse harmonique,*Ann. Insee 15, pp 3–104.

[12] Devroye, L., Gyorfi, L., Lugosi, G. (1996). *A probabilistic Theory of Pattern Recognition*. Springer, New York.

[13] Efron, B. (1979). Bootstrap Methods: *Another look at the Jacknife*. The Annals of Statistics. Vol. 7, No. 1, pp 1-26.

[14] Escabias, M., Aguilera, A., Valderrama, M. (2004). *Principal Components Estimation of Functional Logistic Regression Discussion of Two Different Approaches*. Journal of non Parametric Statistics 16(3-4), pp 365–384.

[15] Ferraty, F., Vieu, P. (2006). *Non Parametric Functional Data Analysis. Theory and Practice*. Springer, New York.

[16] Fisher, R. (1936). The Use of Multiple Measurements in Taxonomic Problems. Annals of Eugenics, Vol 7, pp 179-188.

[17] Freund, Y., Schapire, E. (1997). A decision-theoretic generalization of on-line learning and application to boosting. Journal of Computer and System Sciences, 55(1), pp 119-13.

[18] Geurts, P. (2001). *Patern extraction for time series classification*. Principles of Data Mining and Knowledge Discovery, LNAI 2168, pp 115-127. Springer-Verlag Berlin.

[19] Geurts, P., Wehenkel, L, (2005). *Segment and combine approach for nonparametric time-series classification.* In Proceedings of the 9th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD).

[20] Ghattas, B., Nerini, D. (2006). *Classifying densities using functional regression trees: Applications in oceanology.* Computational Statistics-Data Analysis. Vol 51, pp 4984-4993.

[21] Ghattas, B., Perera, G. (2004). *A Memory-Restricted Learning Algorithm*, Prépublicatons IML, CNRS.

[22] Giorgino, T. (2009) *Computing and Visualizing Dynamic Time Warping Alignments in R: The dtw Package.* Journal of Statistical Software, 31(7), pp 1-24. http://www.jstatsoft.org/v31/i07/.

[23] Giraldo, R. (2007). *Análisis exploratorio de variables regionalizadas con métodos funcionales.* Revista Colombiana de Estadística Volumen 30 No. 1. pp. 115 a 127.

[24] Hastie, T., Tibshirani, R., Friedman, J. (2001). *The Elements of Statistical Learning.* Springer-Verlag, New York.

[25] Itakura, F. (1975). *Minimun Prediction Residual Principle Applied to Speech Recognition.* IEEE Transacition on Acoustics, Speech, and Signal Processing, Vol ASSP-23, pp 52-72.

[26] Kadous, M. (1999). *Learning comprehensible descriptions of multivariate time series.* Proceeding of the Sixteenth International Conference on Machine Learning, pp 454-463. Morgan Kaufmann, San Francisco.

[27] Kadous, M., Sammut, C. (2005). *Classification of Multivariate Time Series and Structured Data Using Constructive Induction.* Machine Learning, 58, pp 179-216.

[28] Kass, G. (1980). *An Exploraty Technique for Investigating Large Quantities of Categorical Data.* Journal of Applied Statistics, Vol. 29, No 2, pp 119-127.

[29] Keogh, E. (2002). *Exact Indexing of Dynamic Time Warping.* VLDB 02, pp 406-417, Hong Kong, Agosto 20-23.

[30] Kim, S., Park S., Chu, W. (2001). *An index-based approach for similarity search supporting time warping in large sequence databases.* In Proceeding of the 17th international conference on data engineering, pp 607-614.

[31] Lopez de Mántaras, R. (1991). *A Distance-Based Attribute Selection Measure for Decisión Tree Induction.* Machine Learning. Volumen 6, pp 81-92.

[32] Lugosi, G., Nobel, A. (1996). *Consistency of data-driven histogram methods for density estimation and classification.* Annals of Statistics, 24, pp 687-706.

[33] Morimoto, Y., Ishii, H., Morishita, S., (2001). *Efficient Construction of Regression Trees with Range and Region Splitting.* Machine Learning, 45, pp 235–259.

[34] Murthy, S., Kasif, S., Salzberg, S., Beigel, R. (1993). *OC1: Randomized induction of oblique decision trees. Proc National Conf on Artificial Intelligence*, pp 139-172.

[35] Pezulli, S., Silverman, B. (1993). *Some Properties of Smoothed Components Analysis for Functional Data.* Computational Statistics 8. pp 1-16.

[36] Quinlan, R. (1986). *Induction of decision trees.* Machine Learning, 1(1): 81-106.

[37] Quinlan, R. (1993). *C4.5: Programs for Machine Learning,* Morgan Kaufmann, San Mateo.

[38] Ramsay, J. & Dalzell, C. (1991). *Some Tools for Functional Data Analysis*, Journal Royal Statistical Society 53(3) pp 539–572.

[39] Ramsay, J. & Silverman, B. (2005). Functional Data Analysis, Springer.

[40] Ripley, B. (1996) *Pattern Recognition and Neural Networks*. Cambridge University Press, Cambridge. Chapter 7.

[41] R version 2.9.1 (2009-06-26). Copyrighy (C) 2009 The R Foundation for Statistical Computing. http://www.r-project.org/.

[42] Rodríguez, J., Alonso, C. (2004). *Interval and Dynamic Time Warping-based Decision Trees*. In 19th Annual ACM Symposium on Applied Computing, Special Track on Data Mining, 2004.

[43] Rodríguez, J., Alonso, C. (2000). *Time Series Classification by Boosting Interval Based Literals.* Revista Iberoamericana de Inteligencia Artificial, No 11, 2000, pp 2-11.

[44] Rodríguez, J., Alonso, C.. (2004). *Clasificación de Series: Máquinas de Vectores Soporte y Literales basados en Intervalos.* Revista Iberoamericana de Inteligencia Artificial, No 23, 2004, pp 131-138.

[45] Saito, N. (1994). *Local feature extraction and its application using a library of bases*. Phd thesis, Department of Mathematics, Yale University.

[46] Sakoe, H., Chiba, S. (1978). *Dynamic Programming Algorithm Optimization for Spoken Word Recognitión*. IEEE Transacition on Acoustics, Speech, and Signal Processing, ASSP-26, pp 43-49.

[47] Segal, M., (1992). *Tree-structured methods for longitudinal data*. American Statistical Association. Vol 87, No 418, pp 407-418.

[48] Stan, S., Chan, P. (2007). *Toward Accurate Dynamic Time Warping in Linear Time and Space*. Intelligent Data Analysis. Vol 11, Number 5/2007, pp 561-580.

[49] Stone, M., (1974). *Cross-Validatory Chice and Assessment of Statistical Predictions*. Journal of Royal Statistical Society. Series B, Vol 36, No 2, pp 111-147.

[50] Vapnik, V. (1998). *Statistical Learning Theory*. Wiley, New York.

[51] Webb, P., Yohannes, Y. (1999). *Classification and Regresion Trees, Cart. A user manual for identifying indicators of vulnerability to famine and chronic food insecurity*. Microcomputers in Policy Research. Internacional Food Policy Research Institute.

[52] Xi, X., Keogh, E., Shelton, C., Wei, L. (2006). *Fast Time Series Classification Using Numerosity Reduction*. In ICML 06Ñ Proc. of the 23rd international conference on Machine Learning, pp 1033-1040.

[53] Yamada, Y., Suzuki, E., Yokoi, H., Takabayashi, K. (2003). *Decision-tree Induction from Time-series Data Based on a Standard-example Split Test.* Proceeding of the Twentieth International Conference on Machine Learning, ICML, Washington DC.

[54] Yi, B., Jagadish, K., Faloutsos, H. (1998). *Efficient retrieval of similar time sequences under time warping*. In ICDE 98, pp 23-27.

[55] Yu, Y., Lambert, D. (1999). *Fitting trees to functional data: with an application to time-of-day patterns*. J. Comput. Gragh. Statist. 8, pp, 749-762.

[56] Zhang, H., (1998). *Classification Tree for Multiple Binary Responses*. American Statistical Association, Vol. 93.