Diagnosis de Sistemas Dinámicos Mediante el Aprendizaje de Modelos Proposicionales

Departamento de Lenguajes y Sistemas Informáticos

Memoria de la Tesis Doctoral para optar al grado de Doctor en Informática por la por la Universidad de Sevilla presentada por

D. Pedro J. Abad Herrera

Directores:
D. Rafael Martínez Gasca
D. Juan A. Ortega Ramírez

Sevilla, 2007

Don Rafael Martínez y Don Juan A. Ortega, Profesores Titulares de Universidad adscritos al área de Lenguajes y Sistemas Informáticos,

CERTIFICAN QUE

Don Pedro J. Abad Herrera, Ingeniero Informático por la Universidad de Sevilla, ha realizado bajo nuestra supervisión el proyecto de investigación titulado:

Diagnosis de Sistemas Dinámicos Mediante el Aprendizaje de Modelos Proposicionales

Una vez revisado, autorizamos la presentación del mismo como Tesis Doctoral en la Universidad de Sevilla y estimamos oportuna su presentación al tribunal que habrá de valorarlo

Fdo. Pedro J. Abad Herrera
Doctorando

Fdo. Rafael Martínez
Fdo. Juan A. Ortega
Profesores Titulares de Universidad
Área de Lenguajes y Sistemas Informáticos
Sevilla, 2007

Tesis Doctoral parcialmente subvencionada por la Comisión Interministerial de Ciencia y Tecnología con los Proyectos DPI2006-15476-C02-00 y TSI2006-13390-C02-02.

A Lola

Agradecimientos

Voy a intentar expresar, en breves palabras, mi enorme agradecimiento a aquellas personas que han ayudado en el desarrollo de la presente Tesis.

Por supuesto, a mis padres, que me han dado la oportunidad de recorrer el largo camino que hasta aquí conduce con generosidad y sacrificio.

A mis directores de Tesis, Rafael y Juan, por mostrar su confianza en mi para sacar adelante este trabajo, y por su apoyo y empuje en todo momento. Sin duda, a lo largo de estos años hemos forjado una muy buena relación profesional, y lo que es más importante, también personal.

A mi compañero Antonio, por tantísimo trabajo conjunto y tantos años de camino recorrido juntos.

A Pepe, puesto que su punto de vista siempre es una nueva opción.

A Juan y Juanma, por su colaboración desinteresada y tantas horas de consultas soportadas con el mejor de los ánimos.

A todos aquellos compañeros y amigos que han mostrado su interés por la evolución del presente trabajo, y me han animado y apoyado.

Finalmente, a mi mujer, Lola, porque sin su ayuda, apoyo y comprensión los desánimos y dudas hubiesen supuesto un escollo insalvable. Y como no, a mi hijo Jaime, por tantos 'papa ahora no puede'.

Sinceramente, gracias a todos.

Pedro J. Abad

Índice general

Lista de tablas

Lista de figuras

Lista de algoritmos

Capítulo 1

Introducción

La detección y la diagnosis del funcionamiento anómalo de mecanismos y procesos son importantes desde el punto de vista estratégico de las empresas, debido a las demandas económicas y de conservación del medio ambiente que se requieren para permanecer en los mercados competitivos. En parte esto conduce a que sea un campo de investigación muy activo. Los fallos producidos en los componentes y procesos pueden provocar paradas indeseables y deterioro de los sistemas, con el consiguiente aumento de costos y la disminución de la producción. Por tanto para mantener los sistemas en niveles de seguridad, producción y fiabilidad deseados se necesita desarrollar mecanismos que permitan la detección y diagnosis de esos fallos que se producen en los sistemas.

Por su parte, el campo de la *minería de datos*[1] ha proporcionado en los últimos años una gran diversidad de técnicas informáticas, que son capaces de extraer, de forma automática, información relevante de grandes volúmenes de información almacenada. Dichas técnicas pueden descubrir, por ejemplo, el modelo de funcionamiento de un sistema partiendo de los datos de funcionamiento almacenados de dicho sistema.

El presente proyecto de tesis se centra en la aplicación de técnicas de minería de datos para obtener diagnosticadores de fallos de sistemas dinámicos, que sean poco costosos computacionalmente, rápidos y capaces de trabajar en condiciones de monitorización *on-line*.

[1] del inglés *data mining*

1.1. Generalidades

Entendemos por detección de fallos la tarea de determinar, a partir de las observaciones realizadas, cuándo existe un funcionamiento incorrecto del sistema sujeto a observación, y por diagnosis de fallos cuáles son las causas de ese comportamiento incorrecto.

El concepto de Diagnóstico es muy amplio y cubre prácticamente todos los aspectos de la vida cotidiana: medicina, dispositivos electrónicos, procesos químicos, economía, ecosistemas, etc. En el presente trabajo nos centraremos en el campo de diagnosis de los dispositivos físicos, y más concretamente de las actividades industriales, específicamente en sistemas dinámicos.

Los principales objetivos de los sistemas de diagnóstico para los sistemas industriales serán:

- Evitar situaciones de riesgo, tanto para los seres humanos como para el entorno ambiental donde se desarrolla dicho proceso industrial.
- Ayudar al personal de control en el análisis de la información que generan los dispositivos de medida.
- Mejorar la calidad del producto final mediante la detección de fallos en el proceso o en el producto.
- Facilitar la optimización de la marcha del proceso, mediante la detección de fallos que pueden propagarse a otros componentes del sistema o generar productos defectuosos.

Para mejorar y agilizar el proceso de detección de fallos y su posterior diagnóstico es importante que dichos procesos sean rápidos y eficientes, y es necesario para ello automatizar el diagnóstico, entendiendo como tal la utilización de los ordenadores para reducir o eliminar la intervención humana en el proceso de diagnosis.

En el entorno de los procesos industriales, la monitorización debe representar un fiel reflejo del sistema y de las desviaciones que se producen del comportamiento esperado. La diagnosis permite identificar las partes que fallan. Generalmente los sistemas de diagnosis integran la monitorización (suponemos que los sensores

funcionan correctamente) y la diagnosis (detección e identificación de fallos). La diagnosis de fallos determina por qué un sistema diseñado correctamente no esta funcionado como se esperaba. La explicación de dicho comportamiento erróneo, a partir de una observación determinada, es la principal tarea de la diagnosis de fallos.

1.2. Objetivos de la Tesis

El campo de la diagnosis de fallos es un campo de investigación vigente y con un fuerte crecimiento desde hace algún tiempo, debido a las fuertes necesidades que desde diversos campos de la ciencia y la ingeniería surgen cada día.

El objetivo fundamental que ha orientado la realización de los trabajos que se incluyen en esta memoria ha sido la definición de una metodología, que de una forma automática pueda generar clasificadores de sistemas dinámicos que puedan trabajar *on-line*, y que puedan evaluar el comportamiento del sistema a diagnosticar con unos requerimientos computacionales muy contenidos.

Para conseguir dicho objetivo, nos hemos centrado en las técnicas de $\mathcal{KDD}$[2], y más concretamente en la *clasificación automática*. Dentro de dicha técnica, los *árboles de decisión* son una forma de representar el conocimiento que requiere poca capacidad de almacenamiento, y cuya evaluación puede ser realizada sin grandes exigencias computacionales en muy poco tiempo. Todo ello los hace ideales para conseguir el objetivo pretendido. Los modelos generados son *comprensibles*, ya que pueden ser interpretados por los seres humanos y *proposicionales*, al aprender de una única tabla sin relacionar más de una fila o atributo cada vez [Hernández Orallo *et al.*, 2004].

En la búsqueda de dicho objetivo, se han adaptado las fases del proceso de $\mathcal{KDD}$ a la metodología diseñada con el fin de obtener los resultados deseados. Durante esta adaptación, han surgido diversas necesidades, cuya solución se constituye en objetivos parciales.

Uno de los primeros problemas que hay que solventar es el manejo de grandes cantidades de información provenientes de las lecturas de los sensores del sistema a diagnosticar. Para ello se propone la reducción de la información no relevante

[2] *Knowledge Discovery Databases*

mediante un algoritmo de segmentación conocido, pero esto conlleva que el resultado de la reducción no sea apropiado para realizar aprendizaje supervisado. Otro de los objetivos es por tanto, conseguir que la información reducida vuelva a tener la estructura necesaria para poder aplicar la técnica elegida. Esto se consigue mediante un algoritmo de normalización, que vuelve a dar a la información recogida la estructura que se requiere.

Cuando los patrones existentes en los datos de entrenamiento no permiten una clasificación correcta es necesario buscar estrategias que permitan minimizar el error obtenido en la clasificación. Cuando los datos que se clasifican pertenecen a distintos comportamientos de fallo de un sistema se dice que el sistema tiene problemas de diagnosticabilidad. Uno de los objetivos que se pretenden cubrir es dar alternativas al problema de la diagnosticabilidad, ofreciendo otras opciones cuando dicha situación se presenta.

Además se plantea la necesidad de crear un algoritmo eficiente, que sea capaz de integrar todas las conclusiones obtenidas en los ensayos, y que incorpore aquellas técnicas que demuestren su validez, dando soluciones alternativas a los problemas surgidos y no suficientemente resueltos.

Finalmente todos estos diferentes trabajos se integran de una manera natural en la metodología que se propone y se implementa una herramienta para validar las técnicas desarrolladas sobre sistemas simulados y sobre sistemas reales.

1.3. Marco de la Tesis

Esta tesis doctoral se ha desarrollado en el marco de los siguientes proyectos de investigación:

- **Desarrollo de herramientas basadas en modelos semicualitativos para el análisis de sistemas dinámicos. Aplicación a la supervisión, control y predicción de comportamiento (CICYT** $DPI2000-0666-C02-02$**).** Dentro del proyecto, la tarea relacionada con esta tesis consistía en el estudio de la posibilidad de realizar diagnosis de sistemas dinámicos utilizando técnicas de aprendizaje automático. Para ello, se tomó como referencia un modelo simplificado de la planta para la obtención de datos del sistema. Estos datos se trataron convenientemente antes de aplicar las

herramientas de aprendizaje, dando como resultado un conjunto de reglas que permitieron, de forma aceptable, el diagnóstico de los fallos previstos. Dicho proyecto dio como resultado la base de un conjunto de técnicas para realizar el diagnóstico.

Investigador principal: Rafael Martínez Gasca

Período de duración: $2000 - 2003$

- **Automatización de la detección y diagnosis de fallos de sistemas estáticos y dinámicos usando conocimiento semicualitativo (CICYT** $DPI2003-07146-C02-01$**).** Este proyecto se presenta como continuación del anterior, con el objetivo de profundizar en las técnicas que se desarrollaron en el proyecto anterior, mejorando su eficiencia, solucionando los problemas planteados y llevando su aplicación a sistemas reales. Para ello, se modela en detalle la planta que se utilizó en el modelo previo, y se crean las herramientas que permiten la automatización de las distintas técnicas que abarcan el proceso de diagnóstico. Finalmente dichas herramientas se aplican a la planta real, lo que valida las herramientas desarrolladas y corroboran las expectativas previstas y las posibilidades planteadas.

  Investigador principal: Rafael Martínez Gasca

  Período de duración: $2003 - 2006$

- **Detección automática de fallos, diagnosis y tolerancia a fallos en sistemas con incertidumbre y distribuidos. (CICYT** $DPI2006 - 15476 - C02 - 00$**).** Este proyecto aborda entre sus tareas el tratamiento de los datos relativos a la diagnosis cuando el sistema está distribuido tanto semánticamente como estructuralmente. Se aborda, pues, de esta forma, una nueva perspectiva desde el punto de vista del diagnóstico, donde será necesario adaptar las técnicas desarrolladas a la nueva situación y/o generar nuevas alternativas.

  Investigador principal: Rafael Martínez Gasca

  Período de duración: $2006 - 2009$

1.4. Estructura de la Tesis

El resto del documento se organiza como sigue:

Capítulo 2: Planteamiento del Problema. En este capítulo se encuadra la investigación desarrollada en su entorno, presentando de manera general el problema planteado así como se realiza un esbozo de la solución aportada.

Capítulo 3: Técnicas de Diagnosis Automática. En este capítulo se presentan las distintas aproximaciones, que han ido apareciendo desde los comienzos, para llevar a cabo la diagnosis de los sistemas. Dichas técnicas se han dividido en 4 grandes grupos según el enfoque que se ofrece de la diagnosis. De esta forma tendremos: *la diagnosis basada en conocimiento, diagnosis basada en casos, diagnosis basada en modelos y diagnosis basada en aprendizaje*. Dentro de estos grupos quizás sea la diagnosis basada en modelos la de más amplia difusión y en la que más trabajos aparecen desde los más diversos enfoques. Todos estos enfoques serán tenidos en cuenta, y así podemos encontrar desde la forma de representar el espacio de valores: *cualitativo, cuantitativo o mixto*, hasta las diversas formas de razonamiento, pasando por la representación de la causalidad, etc.

Capítulo 4: Extracción de Conocimiento a Partir de Bases de Datos. Este capítulo presenta el proceso de extracción de conocimiento en bases de datos, conocido más comúnmente como $\mathcal{KDD}$. Se presentan sus distintas fases así como los diferentes campos de aplicación y la diversidad de técnicas que permiten la extracción del conocimiento a partir de los datos en sus distintas vertientes.

Capítulo 5: Características de la Clasificación Automática. Este capítulo versa sobre las características específicas de los problemas de clasificación. Se presentan los motivos que nos llevan a clasificar, la estructura de los problemas, las mejoras que pueden obtenerse y la forma de validar los resultados obtenidos.

Capítulo 6: Diagnosis Mediante Aprendizaje de Modelos Proposicionales. El objetivo del presente capítulo es presentar las aproximaciones realizadas por el autor con objeto de realizar *Diagnosis de Sistemas Dinámicos*

usando Técnicas de Aprendizaje Supervisado que dan como resultado un modelo proposicional del mismo. En este capítulo se presentan dichas aproximaciones, desde la metodología base hasta las mejoras en el tratamiento de la información o las soluciones aportadas al problema de la *diagnosticabilidad*.

Capítulo 7: Casos de Estudio. En este capítulo se validan las técnicas presentadas en el capítulo previo sobre sistemas dinámicos. En concreto, el sistema elegido para dicha validación es un sistema que incorpora un motor de corriente continua. Se selecciona este sistema por ser un sistema de dinámica muy rápida, que permite comprobar la efectividad de las aproximaciones propuestas. Se comienza con un modelo elemental del mismo hasta llegar a diagnosticar el sistema real.

Capítulo 8: Conclusiones y Trabajo Futuro . Este capítulo recoge las principales conclusiones obtenidas en la elaboración del presente proyecto de tesis, así como las futuras líneas de investigación que pueden surgir a partir de del trabajo ya desarrollado.

Apéndice A: Hardware y Software Desarrollado. Este apéndice se ha incluido para mostrar los detalles del hardware y software que se ha desarrollado durante la evolución del presente proyecto de tesis.

Capítulo 2

Planteamiento del Problema

En este capítulo se realiza una presentación general de la investigación realizada, detallando de manera especial el problema planteado, antecedentes y motivación, así como la solución que se propone.

2.1. Antecedentes

En sus inicios, los dos primeros campos de investigación en diagnóstico automático fueron la medicina (el proyecto MYCIN [Buchanan and Shortliffe, 1984a] comienza en 1972) y el diagnóstico de dispositivos mecánicos y/o electrónicos (conocido en la literatura específica como *troubleshooting*). Por otra parte en los últimos años se ha extendido el estudio de la diagnosis a otros campos como pueden ser los procesos biológicos, la depuración de software o los procesos industriales.

Podemos distinguir, de esta forma, 5 grandes campos de aplicación:

- Medicina, que fue uno de los campos pioneros con el proyecto MYCIN [Buchanan and Shortliffe, 1984a] o el proyecto GASNET/GLAUCOMA [Kulikowski and Weiss., 1982]. En la actualidad, sigue siendo uno de los campos más ampliamente estudiados [Linguraru *et al.*, 2006; Chaisaowong *et al.*, 2007]

- Dispositivos físicos, entendiendo estos como sistemas electrónicos o industriales. [Ng, 1991; Harutunyan *et al.*, 2006]

- Procesos químicos. DISARM [Vinson *et al.*, 1992], [Dash *et al.*, 2003]

- Depuración de Software. [Sabin *et al.*, 1995; Wotawa, 1996; R. *et al.*, 2002; Chen and Wotawa, 2006]
- Sistemas biológicos, que se pueden considerar como procesos biológicos en el ámbito industrial o ecosistemas. [Struss and Heller, 1999; Davids *et al.*, 2006]

Por otra parte la diagnosis puede ser realizada en tres tipos de sistemas distintos, cada uno de ellos con sus propias características:

- Sistemas estáticos, entendiendo como tales aquéllos cuyas salidas dependen únicamente de las entradas y de algunos parámetros constantes del sistema. No dependen del tiempo en su evolución, como pueden ser la diagnosis de componentes electrónicos. [de Kleer *et al.*, 1992; Yvan *et al.*, 2005].
- Sistemas dinámicos, entendiendo como tales aquéllos sistemas donde existe una evolución temporal, y el fallo de algún componente del sistema provoca que su evolución temporal sea diferente. En estos sistemas la salida del sistema depende las variables de entrada, los parámetros del sistema y el instante en el que se encuentre, lo que conforma el estado del sistema. El comportamiento del sistema puede entonces ser descrito en términos de evolución de un estado a otro. [Struss, 1997; Liu *et al.*, 2006].
- Sistemas híbridos. (Discrete Event System). Los sistemas híbridos son sistemas dinámicos complejos, cuyo comportamiento se modela como un sistema híbrido. Los modelos híbridos presentan tanto comportamientos discretos como continuos. Son representados típicamente como una secuencia de trozos de comportamiento continuo intercalados con transiciones discretas. Cada periodo de comportamiento continuo representa lo que se llama modo del sistema. Las transiciones entre modos dan como resultado cambios en el comportamiento continuo del sistema así como el vector de estado que inicializa el comportamiento en el nuevo modo. Las transiciones discretas son modeladas como autómatas de estados finitos, lógica temporal, funciones de cambio o algún otro sistema de transición. El comportamiento continuo dentro de un modo del sistema se modela con ecuaciones diferenciales y algebraicas u otros métodos. [McIlraith *et al.*, 1999; García *et al.*, 2005].

Cuando hablamos además de sistemas dinámicos, es necesario tener en cuenta que debido a la dinámica del sistema los fallos que se pueden detectar son de tres tipos:

- Fallos abruptos, entendiendo como tal el fallo que se produce de forma instantánea en un momento dado, y a partir de ese instante permanece presente en el sistema en la misma magnitud.
- Fallos incipientes, que son aquéllos fallos que, normalmente debido al desgaste de algún componente del sistema, comienza como un fallo de magnitud pequeña y va evolucionando en el tiempo a una magnitud mayor, hasta alcanzar un máximo.
- Fallos intermitentes, que son los fallos que se producen en el sistema de forma intermitente, remitiendo tras un tiempo y volviendo a producirse pasado un periodo de tiempo. Son los más difíciles de detectar y constituye uno de los principales problemas abiertos en el campo de la diagnosis de sistemas dinámicos.

Esta distinción no ocurre en los sistemas estáticos debido a que en ellos no existe dependencia del tiempo, con cual el fallo está o no presente en el momento de la detección.

Dependiendo del momento de detección del fallo los sistemas de diagnóstico pueden ser:

- *OFF-LINE*, que se produce cuando los requisitos de tiempos entre la ocurrencia del fallo y la detección del mismo no son considerados importantes. En estos casos el diagnosticador puede ser ejecutado a voluntad, sin necesidad de que esté continuamente diagnosticando el sistema. En [Struss *et al.*, 1997] lo encontramos aplicado a la diagnosis del circuito hidráulico del ABS de un vehículo.
- *ON-LINE* o diagnosis a bordo, donde el sistema de diagnóstico debe reaccionar rápidamente ante las anomalías, y ejecutarse con prontitud en función de interpretar dichas anomalías y, en su caso, tomar las acciones de recuperación oportunas. En estos casos el sistema está continuamente siendo monitorizado y diagnosticado ante la posibilidad de un fallo. En [Cascio *et al.*, 1999] se

presenta el uso de la diagnosis basada en modelos y el modelado cualitativo para realizar diagnosis a bordo en el campo de la automoción, concretamente en el sistema de alimentación de combustible *Common Rail*. En [Hutter and Dearden, 2003] se presentan algoritmos cuya eficiencia permite realizar diagnosis *on-line* en sistemas con poca capacidad de cómputo.

Según la forma en la que se realicen las tareas que conforman el proceso de diagnosis tendremos:

- Diagnosis centralizada. Hablamos de diagnosis centralizada cuando las tareas que se realizan durante el proceso de diagnosis se llevan a cabo de forma secuencial, es decir, no se comienza una tarea hasta que no se finaliza la precedente. Este es el caso de la mayoría de las propuesta de diagnosis hasta la actualidad.

- Diagnosis distribuida. En este tipo de diagnosis las tareas que conforman el proceso de diagnosis se realizan en paralelo siempre que sea posible. De esta forma los resultados se obtienen con mayor prontitud, debido al ahorro de tiempo que supone la realización de tareas simultáneamente. Por contra se hace necesario disponer de más de una unidad de procesamiento para la realización de la diagnosis, con lo cual aumenta el coste de la misma.

2.2. Diagnosis de Sistemas Dinámicos

Como se ha comentado con anterioridad la diagnosis puede ser aplicada tanto a sistemas estáticos como a sistemas dinámicos. En un primer momento los esfuerzos estuvieron enfocados en la diagnosis de sistemas estáticos, debido a su menor complejidad, y a partir de esas primeras técnicas, desarrolladas para los sistemas estáticos, se extendió su aplicación a los sistemas dinámicos.

La característica común a este tipo de sistemas es su comportamiento dinámico, donde el tiempo es un factor muy importante a tener en cuenta, y donde la dinámica de los síntomas puede ser comparable a la del propio proceso, de tal forma que no se detectará de la misma forma una pequeña fuga de un tanque que la obstrucción total de una válvula.

Los sistemas dinámicos en general y los procesos continuos en particular disponen de una serie de restricciones que los hacen diferentes desde el punto de vista de la diagnosis:

- Es necesario vigilar la evolución temporal del proceso y comprobar su consistencia con el estado, o conjunto de estados, obtenidos a partir de la simulación de los modelos.
- Los modelos han de incluir controladores en lazo cerrado.
- La presencia de controladores pueden compensar los fallos, que se manifestarían de forma intermitente.
- La dinámica de los sistemas los lleva a moverse a estados estacionarios a través de estados transitorios.
- El deterioro progresivo de un sistema donde no se haya diagnosticado a tiempo un fallo puede provocar fallos posteriores, con lo cual es necesario imponer restricciones temporales en cuanto al tiempo disponible para diagnosticar este tipo de sistemas.

De todo lo anterior podemos obtener las siguientes conclusiones:

- Se necesita dotar al sistema de diagnóstico de capacidades adicionales para la inclusión del tiempo.
- El diagnóstico es más costoso desde el punto de vista computacional, debido a la complejidad adicional que supone la dinámica del sistema, lo que lleva en ciertos casos a la necesidad de realizar simulaciones temporales.
- Es muy importante seleccionar los puntos de medida.
- Los fallos pueden depender temporalmente unos de otros.
- La diagnosis de un sistema dinámico debe ser una tarea incremental.

Consideraremos pues sistemas dinámicos aquéllos cuyo comportamiento depende del tiempo, y en especial a los que tienen estados, donde existe una acumulación de energía o información de estados anteriores que condiciona la respuesta ante los estímulos actuales.

2.2.1. Diagnosis de sistemas dinámicos basado en estados versus basado en simulación

Una de las discusiones más significativas que se han observado en los últimos años versa sobre si es suficiente, para llevar a cabo la diagnosis de los sistemas dinámicos, considerar únicamente los estados del sistema, e ignorar lo que sucede entre ellos, o es mejor realizar la simulación del sistema para tener en cuenta los cambios entre estados. En el primer caso, la *diagnosis basada en estados*, sólo se razona sobre los estados simples del sistema, sin tener en cuenta los cambios de estado, mientras que en el segundo caso, la *diagnosis basada en simulación*, el comportamiento del sistema se trata de forma continua, tanto en los estados como en las transiciones entre ellas.

La diagnosis de sistemas dinámicos requiere comprobar la consistencia de las observaciones sobre el tiempo de los comportamientos modelados para el sistema. Una solución sencilla es simular incrementalmente el modelo a medida que las observaciones cambian, para predecir los estados sucesores inmediatos (diagnosis basada en la simulación) usada en [Dvorak and Kuipers, 1989].

La diagnosis basada en estados [Dressler, 1996] evita la simulación y genera la diagnosis apoyándose en la consistencia de los estados observados y el modelo solamente. [Malik and Struss, 1996] establecen una condición necesaria y suficiente para la equivalencia entre diagnosis basada en estado y simulación sin dar prueba de ello. [Struss, 1997] presenta más tarde que si el sistema se modela con restricciones de estados y *CID constraints*, que son reglas generales acerca de la continuidad, integración y derivadas, entonces la diagnosis basada en comprobar el estado y la simulación conducen a diagnosis equivalentes. Mas específicamente, puesto que la aproximación basada en estados no usa la relación entre x and dx/dt, y ambas x y dx/dt deberían ser observables para tener bastante redundancia en la tarea de diagnosis. Algunas veces ambas x and dx/dt no son modeladas en el modelo de simulación o no pueden medirse ambas como variables físicas. Ciertos trabajos introducen pseudo-variables para describir el comportamiento dinámico y proporcionar la redundancia suficiente para el diagnóstico del sistema dinámico. En la diagnosis basada en estados, la relación entre el paso de tiempo no se considera, por ejemplo, no se calcula dx/dt a partir de x, o si ambas x y dx/dt son conocidas, se ignora la relación entre ellas.

Los defensores de la propuesta basada en estados [Malik and Struss, 1996; Dressler, 1996],arguyen que la simulación numérica no es aplicable si sólo tenemos información parcial o cualitativa del sistema y sus condiciones iniciales. En estos casos la simulación puede llegar a ser muy compleja, debido a la ambigüedad existente en el conjunto de predicciones de comportamiento. En esos casos la diagnosis basada en estados puede alcanzar buenos resultados sin la necesidad de recurrir a la simulación numérica o cualitativa. En concreto en [Malik and Struss, 1996] se presenta una aproximación basada en consistencia y centrada en los estados del sistema para realizar la diagnosis de un sistema dinámico. Otra aproximación tal como la de [Dressler, 1996] evita la simulación y genera candidatos de diagnostico basados en comprobar la consistencia del modelo con los estados observados solamente (como opuesto a los comportamientos observados, o sea a la secuencia de estados), sin embargo falta un análisis formal de sus precondiciones y de sus consecuencias.

En [Struss, 1997] se presentan los fundamentos de las diagnosis basada en modelos para sistemas dinámicos. En este trabajo se discuten los fundamentos teóricos y los aspectos prácticos de la aplicación de la diagnosis basada en modelos (particularmente la diagnosis basada en consistencia) para sistemas dinámicos. La mayoría de las aproximaciones presentadas requieren la simulación del sistema a ser diagnosticado. En este trabajo se presentan las condiciones para evitar el paso de la simulación, que es con frecuencia prohibitivo y costoso. Una vez que se tuviese la simulación, habría que realizar una comprobación entre la satisfacción del comportamiento y el esperado en ese instante de tiempo. Esta idea subyace en varias aproximaciones (MIMIC [Dvorak and Kuipers, 1989]) y se mantiene, incluso, cuando la simulación es cualitativa y al haber muchos resultados ambiguos, para identificar el fallo se requiere la simulación de muchos escenarios de fallos.

Un trabajo posterior [Panati and Drupé, 2000] presenta unos resultados opuestos a los presentados [Malik and Struss, 1996; Struss, 1997] para el mismo ejemplo, mostrando que la simulación puede ser útil para restringir el uso de posibles diagnósticos. También discute como los resultados de la simulación basada en estados en contextos diferentes no produce los mismos resultados y muestra como se puede razonar a partir de la causalidad del sistema y usando la simulación. No obstante reconoce la mayor eficiencia de la aproximación basada en estados, al no tener que

realizar la búsqueda en el espacio de estados, aunque en [Panati *et al.*, 2000] se muestra que el uso de dependencias causales en la simulación puede hacer factible dicha búsqueda.

2.3. Hipótesis para la solución del problema

Existen un gran variedad de técnicas aplicadas al campo de la diagnosis. Concretamente en la línea que nos ocupa, la diagnosis de sistemas dinámicos, se han presentado una gran variedad de aproximaciones, desde muy distintos puntos de vista. Quizás, el área más profusamente usada ha sida la diagnosis basada en modelos. En el Capítulo 3 se hace una revisión de todas estas aproximaciones.

Nuestra solución se presenta como una metodología compuesta de dos fases claramente diferenciadas.

La primera de las fases se desarrolla *off-line*, y es previa al proceso de diagnóstico propiamente dicho. En esta fase se parte de un conjunto de trayectorias generadas por el sistema a diagnosticar. Las trayectorias representan la evolución del sistema en el transitorio en el que se pretende diagnosticar el sistema. Las trayectorias podrán ser generadas mediante manipulación del propio sistema o, si esa opción no es posible, mediante simulaciones de un modelo representativo del sistema.

Una vez capturadas, o generadas, dichas trayectorias serán almacenadas en una base de datos que contendrán las trayectorias recogidas junto con una etiqueta que indicará el estado en el que estaba el sistema cuando se generaron (algún fallo concreto o la ausencia de él).

Estas trayectorias serán tratadas con objeto de reducir la dimensión de las mismas y añadir nuevos atributos que puedan ser representativos de los fallos que representan las trayectorias.

Finalmente se aplicará algún método de clasificación automática que permita descubrir el modelo proposicional inherente a la información suministrada. Posteriormente, el modelo es validado y si se considera adecuado, se implanta como modulo de diagnóstico del sistema. De no considerarse válido el modelo proposicional, se procede a un ajuste de parámetros o una nueva captura de datos hasta conseguir la adecuación requerida.

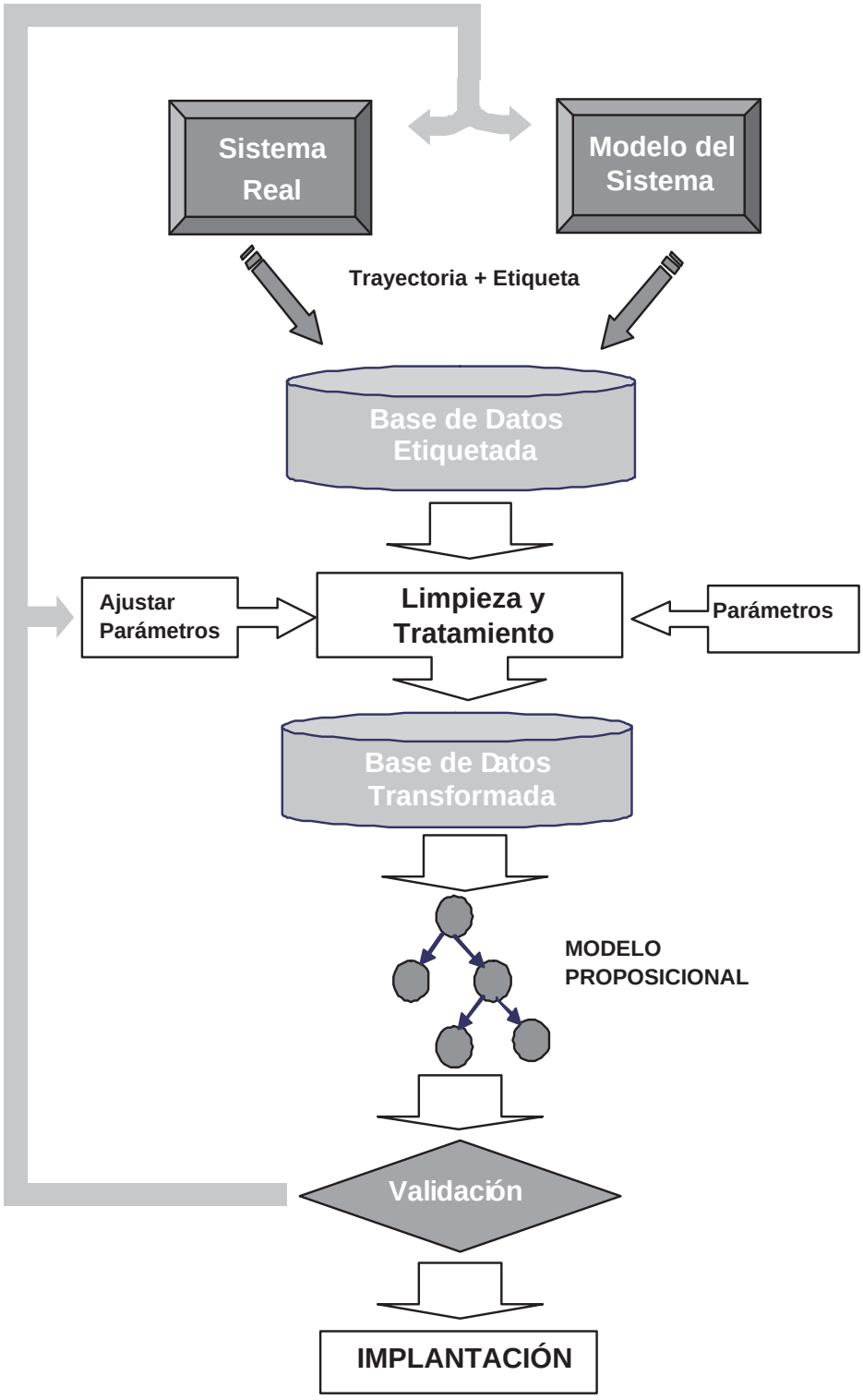

Figura 2.1: Fase *Off-line* de la metodología general

Con esto finalizará la fase *off-line*, que tiene como resultado el modelo proposicional descubierto a partir de las trayectorias proporcionadas. En la figura 2.1 se muestran, de forma general, los pasos de dicha fase.

Una vez conseguido el modelo proposicional, la fase *on-line* de la metodología se desarrolla mientras el sistema está siendo monitorizado, y es la fase de diagnosis propiamente dicha. Para alcanzar el diagnostico, tan sólo hay que evaluar la nueva observación con el modelo proposicional conseguido de la fase previa, tal como muestra la figura 2.2.

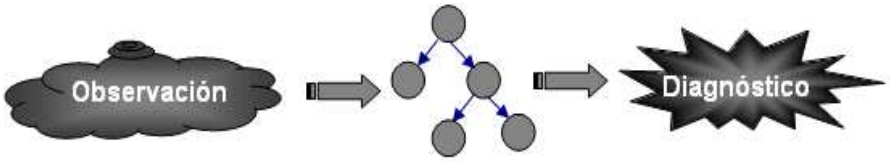

Figura 2.2: Fase *On-line* de la metodología general

En el capítulo 6 se describe en detalle la solución propuesta.

Capítulo 3

Técnicas de Diagnosis Automática

A continuación expondremos las diferentes técnicas que se utilizan a la hora de realizar la diagnosis, según se represente el conocimiento que se tiene del sistema a diagnosticar. Dicha representación proporcionará la relación existente entre los síntomas observados y la diagnosis obtenida dependiendo de la representación utilizada.

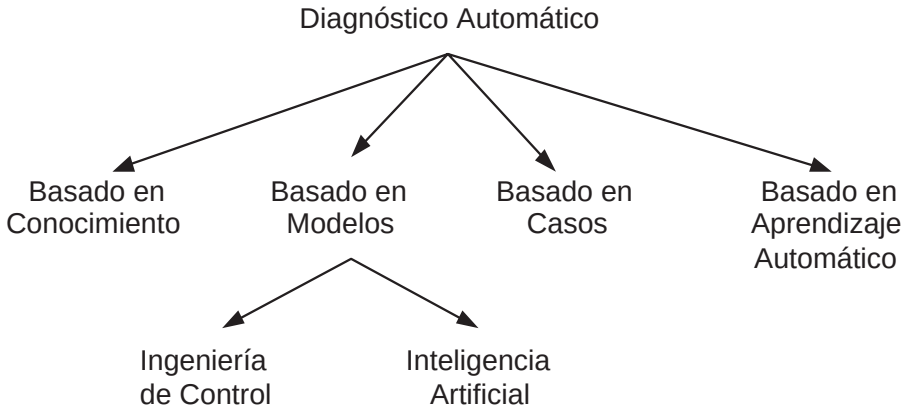

Figura 3.1: Técnicas para la diagnosis automática

Desde el punto de vista de cómo podemos conocer la relación entre los síntomas observados y su diagnosis, como se representa en el sistema dicha relación y como se usa dicha representación para diagnosticar fallos; Balakrishnan y Honavar [Balakrishnan and Jonavar, 1998] distinguen las siguientes técnicas de diagnosis:

- Técnicas basadas en conocimiento.

- Técnicas basadas en casos.
- Técnicas basadas en modelos.
- Técnicas basadas en aprendizaje automático

Las posibilidades en el campo del diagnóstico automático están representadas en la figura 3.1.

3.1. Diagnosis basada en conocimiento

En la diagnosis basada en conocimiento, el conocimiento está basado en los manuales de mantenimiento y la interacción con los expertos para obtener el conocimiento heurístico acerca del mantenimiento y reparación de los dispositivos o procesos. Es habitual representar el conocimiento como reglas o estructuras, que se organizan dentro de una jerarquía de problemas tal y como se representa en la figura 3.2. En el nivel más alto de la jerarquía está el conocimiento general que representa un problema con el dispositivo o proceso. Este problema es refinado sistemáticamente hasta los nodos terminales de la jerarquía, que representan reparaciones físicas o ajustes físicos del dispositivo. Después de que un técnico realice las reparaciones algunos sistemas comprueban que el fallo diagnosticado haya desaparecido mediante el *backtracking* en la jerarquía.

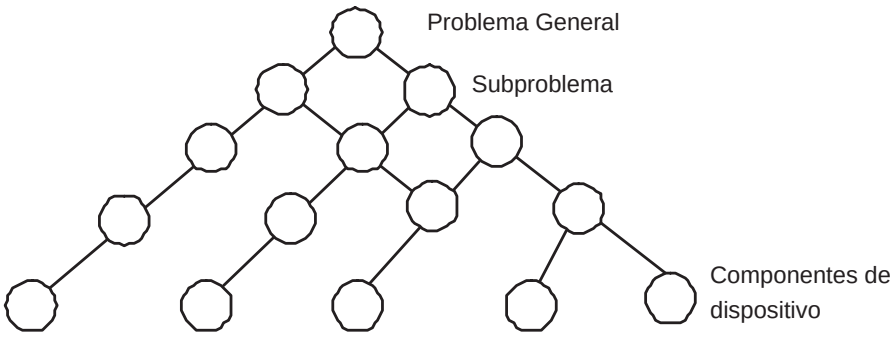

Figura 3.2: Jerarquía de diagnostico

Estos sistemas usan la experiencia adquirida en el dominio a diagnosticar, que es muy útil cuando no existe un modelo del sistema o es difícil de obtener. La técnica más significativa en este campo es la utilización de los Sistemas Expertos,

que permiten representar el conocimiento de múltiples formas (mediante reglas, grafos y/o, etc.) En [Chittaro *et al.*, 1992; Tezafestas, 1987] tenemos recopilaciones al respecto.

Un campo extensamente usado en este tipo de diagnosticadores ha sido el de la medicina, creándose sistemas expertos para diagnosticar las distintas dolencias de los pacientes según las especialidades médicas. MYCIN [Buchanan and Shortliffe, 1984b] desarrollado entre 1972-1980 fue el pionero en este tipo de sistemas, y se aplicó al tratamiento de las infecciones microbianas. MYCIN es un programa interactivo que diagnostica ciertas enfermedades infecciosas, prescribe la terapia antimicrobial, y puede explicar su razonamiento en detalle. En una prueba controlada, su rendimiento igualó la actuación que de los especialistas. Además, el programa de MYCIN incorporó varios desarrollos de IA importantes. MYCIN extendió la noción de que la base de conocimiento debe estar separada del motor de inferencia, y su motor de inferencia basado en reglas se construyó con una estrategia de control de encadenamiento hacia atrás, o meta-dirigido. Desde que se diseñó, como un consultor para médicos, MYCIN tuvo la habilidad de explicar tanto su línea de razonamiento como su conocimiento. Debido al rápido desarrollo en el campo de la medicina, la base de conocimiento se diseñó para ser ampliada fácilmente. Teniendo en cuenta que el diagnóstico médico tiene a menudo un grado de incertidumbre, las reglas de MYCIN incorporaron ciertos factores para indicar la importancia (es decir, probabilidad y riesgo) de una conclusión.

Aunque MYCIN nunca se usó rutinariamente, ha servido de inspiración a otros sistemas que lo siguieron como EMYCIN [Melle, 1980], con un núcleo basado en MYCIN, o NEOMYCIN [Clancey, 1984], que tuvo una base de datos extendida, NEOANEMIA [Lanzola *et al.*, 1990] es un sistema experto creado para diagnosticar las posibles causas de la anemia, GASNET/GLAUCOMA [Kulikowski and Weiss., 1982], INTERNIST [Pople, 1975] ONCOCIN [Shortliffe *et al.*, 1981] y ROGET [Bennett, 1985] entre muchos otros.

En este tipo de sistemas, el conocimiento que se posee del mundo real esta representado de forma que es independiente del método usado para extraer información de él, y normalmente proviene de la observación directa o de la información acumulada por los expertos a través de su experiencia.

En [Heiming and Lunze, 1997] podemos encontrar una aproximación a la diagnosis basada en conocimiento, que usa la estructura causal del sistema real para reducir la complejidad del algoritmo de diagnosis. En esta aproximación se usa información cualitativa, que es obtenida al dividir el valor continuo de las señales en intervalos. Una vez se tiene la información cualitativa, junto con la estructura causal obtenida a partir de la observación, se representa lógicamente la relación existente entre las entradas y las salidas. Para encontrar la diagnosis, a partir de los síntomas observados, se usan métodos de lógica y teoría de probabilidad, todo ello sobre un ordenador con multiprocesamiento y computación paralela.

En el campo de los dispositivos electrónicos, el sistema DART [Genesereth, 1984] presenta un diagnosticador automático independiente de los dispositivos, que trabaja directamente a partir de las descripciones de diseño, usando un lenguaje lógico independiente del dispositivo para describir los dispositivos, y un procedimiento de inferencia sobre dicho lenguaje para llevar a cabo la diagnosis.

Aunque los Sistemas Expertos han siendo ampliamente utilizados, adolecen de unos inconvenientes que actualmente no han sido eliminados en su totalidad: la dificultad en la adquisición y representación del conocimiento experto, la carencia de una metodología de desarrollo universalmente aceptada, los problemas de mantenimiento ocasionados por su fragilidad debida a la falta de estructuración de su conocimiento y, posiblemente la más importante, la dependencia del dispositivo. Existen trabajos encaminados a solucionar estos problemas [Breuker and de Velde, 1994; Guida and Tasso, 1995], pero el principal escollo es que la solución depende del problema para el que fue construido. No obstante, si el problema está bien definido y acotado, los Sistemas Expertos siguen siendo una herramienta muy potente de diagnóstico y siguen siendo utilizados. En [Yu *et al.*, 2003] se recurre a ellos como herramienta para la diagnosis en tiempo real de procesos químicos.

3.2. Diagnosis basada en casos

Un razonador basado en casos resuelve nuevos problemas por adaptar las soluciones que fueron usadas para resolver antiguos problemas [Riesbeck and Schank, 1989]. La intuición detrás del Razonamiento Basado en Casos es usar el conocimiento ganado al resolver problemas similares en el pasado como punto de co-

mienzo desde los cuales se resuelven problemas actuales. Como generalmente hay algunas diferencias entre el problema objetivo (entrada) y el caso o casos recuperados, puede ser necesaria algún tipo de modificación para ajustarse a la nueva situación. Este proceso se denomina adaptación. Los casos que fueron exitosos pueden ser guardados para nuevas situaciones.

Un sistema de *Razonamiento Basado en Casos* (CBR[1]) almacena los episodios de problemas pasados como casos que más tarde pueden ser recuperados y usados para ayudar a resolver un nuevo problema. *CBR* se basa en dos observaciones acerca de la naturaleza del mundo: que el mundo es regular, y por tanto problemas similares tienen similares soluciones, y que los tipos de problemas encontrados tienden a ser recurrentes. Cuando estas dos observaciones se sostienen, es conveniente usar este tipo de razonamiento.

El proceso de un razonador de este tipo comprende cuatro pasos [Aamodt and Plaza, 1994] de forma cíclica:

- **Recuperar** los casos más similares.
- **Reusar** los casos para resolver los problemas.
- **Revisar** la solución si es necesario.
- **Retener** las nuevas soluciones como un caso nuevo.

La aplicación de esto a problemas reales levanta un conjunto de cuestiones respecto a los dominios de aplicación. Estas cuestiones incluyen representación de los casos, indexado, almacenamiento, método de recuperación y método de adaptación.

- La representación de los casos considera qué información debe contener. Un caso puede considerarse como una historia para aprender, como el contexto y la solución del problema, o también como un proceso por el cual el problema se resuelve.
- El contexto de un caso se describe por los índices de los casos, que describen bajo que circunstancias es apropiado almacenar un caso.

[1]Del inglés Case Base Reasonig

- El almacenamiento de casos se centra en guardar aquellos casos que se han seleccionado para una recuperación posterior.

- Los métodos de recuperación proporcionan el medio por el cual extraemos aquellos casos del caso base que más cerradamente ajusta la descripción del problema nuevo. Los algoritmos de recuperación se basan en los índices de caos y la organización de los casos almacenados para dirigirlos eficientemente hacia casos útiles potenciales.

- La adaptación es el método por el cual los casos se cambian en algo que es útil para resolver el nuevo problema. Tres métodos bien conocidos de adaptación son: *analogía derivacional, sustitución y transformación.* En la analogía derivacional, se obtiene una solución nueva usando el mismo método por el que fue calculada la solución vieja. Los métodos de sustitución escogen e instalan un reemplazado para algunas partes de la solución vieja que no se ajustaba a los requisitos de la situación actual. Los métodos de transformación usan heurísticas para reemplazar, borrar, o añadir componentes a una solución antigua para hacer que la solución vieja trabaje en la nueva situación.

El razonamiento basado en casos ha sido aplicado satisfactoriamente a múltiples dominios, que van desde tareas de negocios hasta diseños de arquitectura. Para hacer CBR más flexible, los diseñadores han combinado CBR con otras técnicas de IA como pueden ser: *algoritmos genéticos, razonamiento basado en modelos, técnicas de aprendizaje...* En el campo de la diagnosis el razonamiento basado en casos no ha tenido excesiva repercusión, puesto que la diagnosis basada en conocimiento (que es una técnica anterior) puede ser vista como una forma de organización CBR.

La diagnosis basada en casos utiliza el razonamiento basado en casos como forma de discernir la situación en la que se encuentra el sistema, ya sea el comportamiento correcto, o cualquiera de las situaciones de fallos descritas en el conocimiento de los casos.

La diagnosis se realiza mediante la asociación de la nueva situación con uno de los casos previamente conocidos, y ofreciendo como resultado el de dicho caso. En este tipo de técnicas no existe representación explícita del conocimiento ni del

proceso de razonamiento con el que se obtiene la solución. En [Lim *et al.*, 1996; Nakatami *et al.*, 1996] tenemos ejemplos de este tipo de razonamiento

Por citar algunos trabajos dentro del campo de la diagnosis con CBR podemos destacar:

- CASEY [Koton, 1989], que es un sistema para diagnosticar fallos cardiacos. Usa como entradas los síntomas de los pacientes, con los que genera una red causal de los posibles estados internos que podrían producir dichos síntomas. Cuando surge un nuevo caso CASEY intenta encontrar casos de pacientes con síntomas similares, aunque no necesariamente idénticos. Si el nuevo caso coincide entonces CASEY adapta la diagnosis recuperada considerando las diferencias entre los síntomas del caso antiguo y el caso nuevo.

- PROTOS [Porter and Bareiss, 1987], que se desarrollo para problemas auditivos. Aprende a clasificar los problemas auditivos basándose en descripciones de los síntomas de los pacientes, historial y resultados de tests. PROTOS se entrenó con 200 casos agrupados en 24 categorías, tras lo cual alcanzó una precisión del 100

- CASCADE [Simoudis, 1992], diagnostica las causas de fallo del sistema operativo VMS y sugiere una solución. Aunque es simplemente un sistema de recuperación de casos y sugerencias (no es adaptativo), ayuda a una recuperación efectiva en las caídas del sistema.

La principal ventaja de los sistemas de Razonamiento basado en casos frente a los Sistemas Expertos es que no necesita extraer ni representar de forma explícita el conocimiento del que se dispone. Sin embargo, las soluciones que aportan dependen totalmente del problema a resolver. Además, no se explica el mecanismo de razonamiento.

En [Sqalli and Freuder, 1998] se usa el razonamiento basado en casos junto con la satisfacción de restricciones para el testeo de interoperatividad en redes ATM. En este caso el sistema se modela como un CSP (Constraint Satisfaction Problem) y el razonamiento basado en casos (CBR) soporta el proceso de aprendizaje a través del suministro de nuevos casos que pueden en el futuro servir para solucionar casos similares. El CBR es usado también para actualizar el modelo CSP y hacerlo más robusto con objeto de poder solucionar más problemas.

3.3. Diagnosis basada en modelos

La diagnosis basada en modelos razona a partir de un modelo que representa de forma explícita el sistema a diagnosticar. Si el comportamiento de la situación observada no se ajusta a la estimación realizada por el modelo para dicha situación se concluye que hay un fallo, y en un análisis posterior de las diferencias se intenta identificar el componente concreto del fallo. En [Hamscher, 1991; Thomas, 2002] tenemos casos para esta técnica.

La mayoría de las aproximaciones aparecidas en la última década para realizar diagnosis se han basado en el uso de modelos (DBM [2]). Estos modelos se apoyan en el conocimiento del sistema a diagnosticar, que puede estar bien estructurado formalmente y de acuerdo con teorías bien conocidas o puede ser conocido a través de la experiencia de un experto y datos del sistema o proceso. También se presentan algunas veces combinación de ambos tipos de información.

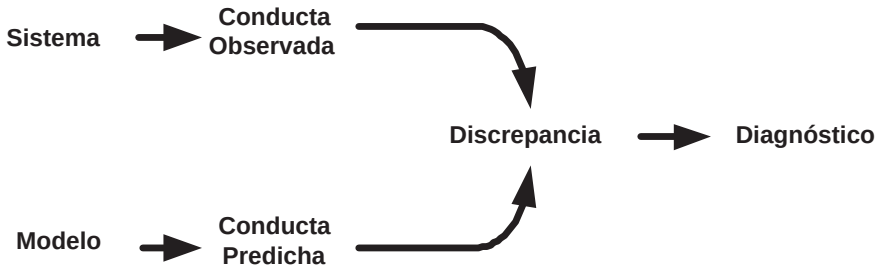

Figura 3.3: Metodología de la Diagnosis Basada en Modelos

La DBM se basa en la comparación de las observaciones que se tienen sobre el funcionamiento de un dispositivo y las predicciones hechas por un modelo del dispositivo sobre su funcionamiento y se resume en la figura 3.3. Las observaciones indican como se comporta el dispositivo, mientras que el modelo nos dice como debería hacerlo si funcionase normalmente. Por tanto si existe una discrepancia entre las observaciones y las predicciones podemos concluir que el dispositivo no está trabajando correctamente.

La información disponible para realizar las predicciones en la DBM será:

- Una descripción local del comportamiento correcto de cada componente.

[2] Diagnosis Basada en Modelos

- Una descripción de la estructura interna del sistema, es decir, la conexión existente entre cada componente del sistema.
- Un conjunto de observaciones, generalmente medidas. Cuando se detecta un síntoma, esto es, se produce una discrepancia entre lo observado y lo predicho, se deduce que alguno de los componentes involucrados en la predicción del valor discrepante de las observaciones funciona de forma incorrecta.

La DBM es independiente del dispositivo, por lo que no es necesaria la acumulación de experiencia sobre un sistema para poder diagnosticarlo.

La diagnosis mediante esta técnica se basa, pues, en un proceso iterativo consistente en:

a. Generación de la hipótesis de diagnóstico. En este paso se enuncia el conjunto de componentes sospechosos en función de las discrepancias observadas.

b. Comprobación de la hipótesis. Consistente en encontrar los candidatos a partir de los sospechosos.

c. Discriminación de la hipótesis. Donde refinaremos el conjunto de candidatos si tras la fase anterior existe más de uno.

En el área de la Inteligencia Artificial un trabajo pionero en este campo fue presentado con objeto de diagnosticar sistemas de componentes basándose en la estructura y su comportamiento [Davis, 1984]. Las primeras implementaciones para llevar a cabo diagnosis fueron DART [Genesereth, 1984] y GDE [DeKleer and Williams, 1987], que utilizaban diferentes mecanismos de inferencia para detectar los posibles fallos. Las formalizaciones de la diagnosis se presentan por [Reiter, 1987; de Kleer *et al.*, 1992], donde se propone una teoría general para el problema de explicar las discrepancias entre los comportamientos observados y correctos de los mecanismos. Apoyándose en ellas, la mayoría de las aproximaciones de diagnosis basadas en modelos para componentes caracterizan la diagnosis de un sistema como una colección de conjuntos mínimos de componentes fallando que explican los comportamientos observados (síntomas). De esto se deriva la importancia de disponer de un buen modelo para hacer diagnosis basada en modelos.

Algunos métodos de diagnóstico usan modelos que requieren que los ingenieros desarrollen *modelos de fallos* además de desarrollar también el *modelo de operación*

normal. Construir *modelos de fallos* en un sistema es útil cuando los fallos son bien conocidos y fáciles de modelar, pero ello limita el sistema de diagnosis a fallos conocidos y no permite ningún tipo de error en el modelado.

Una revisión de las aproximaciones acerca de la automatización de las tareas de diagnosis se puede encontrar en [Dressler and Struss, 1996] y para una discusión de las aplicaciones de la diagnosis basada en modelos se puede consultar [Console and Dressler, 1999]. La generalización de la diagnosis basada en la consistencia se ha propuesto para cubrir sistemas que contienen procesos y que cambian su estructura dinámicamente en [Heller and Struss, 2001].

Los sistemas de diagnostico basados en modelos han llegado a ser claramente exitosos y empiezan a aplicarse a problemas industriales. Muchos, si no la mayoría pueden ser considerados como una variante de General Diagnosis Engine(GDE) siguiendo el principio de diagnosis basada en consistencia [Dressler and Struss, 1996].

No hay que perder de vista de todas formas que razonar en base a un modelo tiene la limitación de que el modelo no es el dispositivo o el sistema, sino una abstracción del mismo, por lo que todos los modelos son necesariamente erróneos, debido a que pueden perder propiedades que sean básicas para el sistema. Por otra parte los modelos estarán condicionados según el fin para el que sean construidos.

Otra desventaja de esta técnica es que no es aplicable donde no se tenga un modelo del sistema a diagnosticar. En esos casos es más aconsejable recurrir a la diagnosis basada en casos o a las técnicas de aprendizaje automático. En [Travé-Massuyes and Milne, 1998] podemos encontrar los problemas con los que se encuentran estas técnicas al ser aplicados a problemas industriales.

3.3.1. Comunidades FDI y DX

Desde hace más de una década se ha manifestado un interés creciente en la comunidad científica por la diagnosis de sistemas físicos. Dos comunidades científicas diferentes han desarrollado dos aproximaciones diferentes. Por una parte la comunidad bajo el nombre de *Identificación y Detección de Fallos*(FDI) que utiliza técnicas procedentes de la teoría de control y análisis estadístico. De entre ellas pueden destacarse la *estimación de estados* [Frank, 1987], *estimación de parámetros* [Isermann, 1993] y la técnica de *redundancia analítica* [Gertler, 1991]. En los

últimos años han aparecido diferentes monografías sobre el tema [Frank., 1996], [Isermann, 1997] y [Patton *et al.*, 2000]. En esta última cita se reflejan algunos de los trabajos desarrollados por la otra comunidad científica, que surgió más recientemente y que se basa principalmente en técnicas derivadas de la Inteligencia Artificial, llamada DX.

Dentro de esta comunidad (DX) y principalmente en la década de los 80 se produce una evolución de los sistemas expertos que hasta entonces llevaban a cabo la diagnosis de los sistemas a una nueva aproximación llamada diagnosis basada en modelos (DBM), [Reiter, 1987], [de Kleer *et al.*, 1992]. El motor de diagnosis general, GDE [DeKleer and Williams, 1987] determinó un marco de trabajo para la diagnosis, que subsecuentemente influenciará casi todos los trabajos en el campo de la diagnosis basada en modelos. El dominio de aplicación principal de este tipo de diagnosis era sobre sistemas estáticos, esto es, sistemas que no evolucionaban en el tiempo.

Los primeros trabajos que aparecen sobre sistemas que evolucionan en el tiempo extienden los algoritmos de los sistemas estáticos (GDE) y consideran QSIM [Kuipers, 1986] como herramienta para detectar los conflictos en estos sistemas [Ng, 1990; 1991]. En trabajos posteriores se ha seguido utilizando QSIM y se ha mostrado [Dressler and Freitag, 1994] como ATMS[3] sirve como un elemento para tratar computacionalmente etiquetas temporales y permite integrarlas con las etiquetas lógicas. También se ha utilizado el conocimiento de las consecuencias de los diferentes modos de comportamiento de los componentes y un modelo de evolución de tales modos [Console *et al.*, 1994]. Puede encontrarse una caracterización general para la diagnosis basada en modelos teniendo en cuenta el tiempo en [Brusoni *et al.*, 1998]. Otras aproximaciones han usado los grafos causales para llevar a cabo la diagnosis [Bousson and Trave-Massuyes, 1992] y los grafos causales temporales [Mosterman, 1997].

Para una revisión de las técnicas desarrolladas en el área de la Inteligencia Artificial puede consultarse [Price, 1999]. Las tendencias sobre la diagnosis basada en modelos y los retos en el futuro se recogen en [Console and Dressler, 1999].

El grupo IMALIA pretende aunar la terminología de ambos enfoques y encontrar las similitudes y complementariedades de cada uno.

[3]Sistemas de mantenimiento de la razón

3.3.2. El problema del modelado

Para experimentación, análisis o diseño, las ciencias y la ingeniería siempre han necesitado representar el mundo real a través de modelos, que pueden ser usados como sustitutos del sistema físico con respecto a las características relevantes para la tarea a realizar.

Un sistema físico puede ser visto desde distintas perspectivas según los fenómenos que tengan lugar en él. En particular debemos distinguir entre modelos estáticos y dinámicos. Si un sistema físico es considerado un conjunto de variables *exógenas* y un conjunto de variables *endógenas*, la evolución del estado del sistema, dado por los valores de las variables *endógenas*, es condicionado por el valor de las variables *exógenas* en el tiempo.

Los modelos estáticos proporcionan una imagen de los estados de equilibrio del sistema, mientras que los modelos dinámicos dan también una descripción de los estados transitorios entre dos estados de equilibrio.

El primer concepto clave de la Diagnosis Basada en Modelos es que el conocimiento acerca de la estructura interna y el comportamiento correcto de un 'artefacto' diseñado puede ser usada para diagnosticar dicho artefacto. La tarea básica de la diagnosis es, así pues, comparar el comportamiento observado del dispositivo con el comportamiento predicho, derivado del modelo del dispositivo, y detectar discrepancias entre ellos. Bajo la asunción de que el modelo es correcto, todas las discrepancias entre la observación y la predicción se toman como fallos del dispositivo. La Diagnosis Basada en Modelos permite realizar la diagnosis de esta forma basándose exclusivamente en:

1. La descripción de la estructura interna del dispositivo, usualmente una lista de los componentes usados y sus conexiones

2. Los modelos de comportamiento correcto de los componentes del dispositivo.

3. Las observaciones

Esta aproximación comenzó con los programas INTER [Kleer, 1976] y SOPHIE [Brown *et al.*, 1982] para el diagnóstico de circuitos electrónicos y DART [Genesereth, 1982] y HT [Davis, 1990] para la diagnosis de circuitos digitales.

El segundo concepto clave de la Diagnosis Basada en Modelos se refiere a las tres tareas fundamentales de la diagnosis: Generación de hipótesis, Comprobación de hipótesis y Discriminación de hipótesis. Reiter [Reiter, 1987] ofrece una teoría general de diagnosis basada en fundamentos lógicos: la diagnosis se define como un conjunto de componentes, de tal forma que, la asunción de que uno de los componentes está fallando, junto con la asunción de que el resto de los componentes del sistema están funcionando correctamente, es consistente con la descripción del sistema y las observaciones. Posteriores trabajos incorporan también modelos de fallo, como por ejemplo SHERLOCK [Kleer and B. C. Williams,].

El tercer concepto clave de la Diagnosis Basada en Modelos es que el modelado es la parte más dura para la DBM. La mayoría de los trabajos para formalizar el proceso de diagnosis y construir algoritmos de diagnosis asumen que se ha dado de antemano la descripción del sistema, tanto estructural como de comportamiento. Cuando se empiezan a aplicar estas técnicas a problemas de la vida real es cuando queda evidente que el modelado es la parte más dura. Es evidente que el Razonamiento Cualitativo puede ofrecer conceptos y técnicas para abordar la tarea del modelado, lo cual explica los fuertes intercambios existentes entre ambas comunidades.

3.3.3. Niveles de abstracción en el modelado

Dependiendo de los aspectos del sistema a diagnosticar el modelado del mismo puede llevarse a cabo mediante la descripción de diferentes aspectos del mismo, en función de la información que quiera ser representada y que resulte útil a la hora de abordar la tarea de diagnóstico. A continuación presentamos las abstracciones más habituales:

Modelado según la estructura y la función. Por estructura entendemos la información acerca de las interconexiones de los elementos que conforman el sistema a modelar. Dicho de otra forma es la información que quedaría al eliminar todas las informaciones textuales de un esquema. Hay dos formas diferentes de organizar dicha información, la funcional y la física. La información funcional de un sistema nos proporciona la organización del sistema de acuerdo a como sus diferentes componentes interactúan, mientras que la física nos indica como están agrupados.

Modelado según el comportamiento. Por comportamiento entendemos la descripción de caja negra de un componente del sistema a modelar, es decir, como la información sale del componente en relación con la información que ha entrado en el mismo. Las técnicas para describir el comportamiento han sido muy variadas, y van desde simples reglas para mapear las entradas y salidas hasta las redes de Petri.

Modelado híbrido. En este tipo de modelado se tienen en cuenta no solamente uno de los aspectos del sistema a modelar como los mencionados anteriormente, sino todos ellos o al menos más de uno. Así en [Davis, 1984] se presenta una forma de razonamiento enfocada a la diagnosis de circuitos electrónicos basada en la estructura y comportamiento de los mismos.

3.3.4. Relaciones Causa-Efecto

Existen dos aproximaciones para representar las relaciones dentro de un modelo. La primera es la aproximación no causal, porque no es necesario representar la dirección específica de la causalidad cuando se modela el sistema. La segunda es la aproximación causal, porque se basa en un modelo causal explícito, generalmente soportado por un grafo orientado [Travé-Massuyes *et al.*, 1993]. No obstante la definición de causalidad relacionado con los fenómenos físicos no parece estar muy clara, aunque la opinión general si parece estar de acuerdo en las tres propiedades siguientes: orden temporal, localidad y necesidad. [Kleer, 1990; Kleer and Brown, 1990b].

No causales Normalmente los ingenieros describen el comportamiento de un sistema en modo estacionario con un modelo algebraico simplificado de la forma $F(X) = 0$, donde X es el vector de variables del modelo. Es posible obtener el comportamiento lineal del sistema en la cercanía de un punto de operación nominal X_0:

$$[\frac{dF}{dX}]_{x0} x dX = 0 \tag{3.1}$$

Los parámetros de este modelo lineal aparecen como derivadas parciales de F.

Causales La representación actual para representar la causalidad es un grafo orientado, llamado Grafo Causal, donde los nodos representan las variables del sistema y los arcos dirigidos las relaciones funcionales entre las variables.

El análisis del sistema consiste en:

1. Identificar las variables relevantes para describir el sistema.
2. Representar sus dependencias causa-efecto como un grafo.
3. Identificar las correspondientes relaciones funcionales.
4. Propagar las influencias a través del grafo.

En la aproximación estática en contra de lo que ocurre en la dinámica, se asume que no hay retraso en la propagación de las influencias.

Híbridos La tarea de supervisión de un proceso físico que no es totalmente conocido significa tener que llevar a cabo distintas tareas que se solapan, como pueden ser predecir y rastrear el comportamiento del proceso, elegir las acciones correctivas adecuadas cuando el proceso no tiene un comportamiento correcto, estimar los comportamientos de las variables no medibles, etc.

El sistema CA-EN [Bousson, 1993; Bousson and Travé-Massuyes, 1993], intenta mimetizar a un operador humano que realice dichas tareas, especialmente aquellas que son por lo general muy tediosas. CA-EN usa como base de tiempo un paso lógico fijo. Dicho paso de tiempo es constante y se establece de acuerdo con la velocidad del proceso. Entre otras cosas esta escala temporal permite evaluar fácilmente la duración de los estados y los instantes de los eventos y ser síncronos con las observaciones provenientes, en tiempo real, del sistema [TravéMassuyes, 1992].

Dos aplicaciones de CA-EN aplicados al campo de la diagnosis son:

- BIOTECH [Steyer, 1991] Es un sistema experto en tiempo real aplicado a la monitorización y diagnosis de un proceso de fermentación. El sistema puede evaluar el estado del proceso desde el punto de vista biológico, detectando, de esta forma, comportamientos anormales y proporcionando avisos permanentes al operador sobre las operaciones a realizar para solucionar la situación crítica.

- TIGER [Milne, 1992; Milne and Travé-Massuyes, 1993].] Tiene como propósito monitorizar y diagnosticar el funcionamiento de una turbina de gas. La herramienta usada es una versión más reciente de predictor CA-EN que la usada en BIOTECH, y que incluye más razonamiento numérico.

3.3.5. Representación del espacio de valores

Modelado Cuantitativo

En estos modelos el sistema es descrito mediante ecuaciones algebraicas que permiten la manipulación numérica del mismo. Estos modelos han sido y siguen siendo usados muy prolíficamente dentro de la comunidad FDI, donde la mayoría de sus aproximaciones se basan en este tipo de modelos, como las ya comentadas *estimación de estados* o *estimación de parámetros*.

Los modelos son ecuaciones diferenciales que describen las dependencias entre las señales de forma cuantitativa, como el modelo del espacio de estados:

$$x = f(x, u, t) \tag{3.2}$$

$$y = g(x, u, t) \tag{3.3}$$

Los fallos son interpretados como señales de entrada externa o desviación de los parámetros. La tarea de diagnosis se solventa a través de la estimación del estado real o la estimación de los valores reales de los parámetros. Estas aproximaciones son útiles si las ecuaciones diferenciales 3.2 y 3.3 son conocidas y las señales son medidas cuantitativamente, es decir, sus valores son valores numéricos exactos

El uso de los modelos cuantitativos, aplicados a la diagnosis, tiene una serie de inconvenientes, pues cuando son usados existe una complejidad del modelado y simulación, además de que si hay imprecisión o incertidumbre se necesita redefinir el concepto de consistencia. Pero sin embargo, estos modelos pueden evitar el problema del bucle realimentado [Dressler and Struss, 1994].

Modelado Cualitativo

La simulación cualitativa persigue derivar descripciones cualitativas del comportamiento a través del tiempo de un sistema a partir de un modelo general del

sistema. Aunque los algoritmos difieren en la forma de representar el comportamiento, todos se basan en las siguientes ideas:

- El dominio de las variables que representan parámetros físicos pueden ser divididos en un pequeño número de marcas e intervalos entre ellas que representan las distinciones cualitativas reales de la magnitud de la variable.
- Conocer la dirección de cambio de la variable, junto a su magnitud cualitativa, es con frecuencia suficiente para determinar su evolución cualitativa.
- Para restringir el comportamiento de las variables es, a menudo, suficiente con conocer algunas de sus propiedades, como puede ser la monotonía.
- El tiempo no es cuantificado a priori. Los puntos de tiempo vienen determinados por 'acontecimientos significativos', como puede ser que una variable pase una de las marcas.

Históricamente hay tres aproximaciones principales al modelado cualitativo y razonamiento sobre sistemas dinámicos:

ENVISION [Kleer and Brown, 1990a] . Creado por De Kleer y Brown es la aproximación basada en componentes. La idea es desarrollar una versión cualitativa del campo de los sistemas dinámicos donde una situación física es vista como un dispositivo compuesto por componentes individuales conectados entre ellos. Se pretende desarrollar una librería de modelos de componentes, genéricas para un dominio dado, cada una con una descripción de comportamiento. Se acuña el termino envisioning para describir el proceso de inferir la descripción del comportamiento de un sistema en términos de sus estados permitidos, el valor de sus variables y la dirección de cambio de dichas variables.

QSIM (Qualitative Simulation) [Kuipers, 1990] . Desarrollado por Kuipers es la aproximación basada en restricciones. El modelo cualitativo del sistema consiste en las variables que componen el sistema en un nivel dado de abstracción y las restricciones que relacionan las variables. Las principales características son la provisión de representación cualitativa para las cantidades, un algoritmo eficiente de simulación cualitativa, un método para

seleccionar transiciones de estados y las cuestiones de completitud y validez de los comportamientos cualitativos.

QPT (Qualitative Process Theory) [Forbus, 1990b] . Elaborado por Forbus es la aproximación basada en procesos. Se tratan tanto los problemas de modelado como de simulación. De las tres aproximaciones es la que tiene el mecanismo de modelado más sofisticado. Como en otras aproximaciones encontramos formas de representar espacios cualitativos, ecuaciones diferenciales y descripción de funciones, pero también una forma de ver colecciones de satisfacciones individuales de condiciones seguras, como agregados con propiedades de comportamiento particulares y una representación explícita para procesos. En [Forbus, 1990a; 1992] Forbus describe el Qualitative Process Engine (QPE), una implementación de QPT, que usa ATMS para hacer *envisioning*.

Algunos sistemas de diagnóstico basado en modelos, donde el modelado cualitativo juega un papel importante son SOPHIE III y DEDALE [Dague P. and Raiman, 1992; Dague, 1994] para la diagnosis de circuitos analógicos. En ellos la diagnosis se entiende como diagnosis estática, los fallos existen desde el principio y no evolucionan durante la diagnosis y no aparecen nuevos fallos

La diagnosis de circuitos electrónicos en modo dinámico ha sido también abordada desde el punto de vista del modelado cualitativo. Así en XDE [Hamscher, 1991] se diagnostican complejos circuitos digitales secuenciales, que son dispositivos dinámicos, más concretamente dispositivos con *estados*, y en CATS/DIANA [Dague, 1994] se diagnostican fallos en circuitos analógicos en modo dinámico.

Pero el modelado cualitativo no se ha aplicado únicamente a la diagnosis de circuitos electrónicos. Para solucionar el problema de la falta de modelo del sistema muchos autores están recurriendo al uso del modelado cualitativo y la programación con restricciones para casos en los que no existe modelo del sistema o el sistema es excesivamente complejo. Un enfoque de aplicación de modelos cualitativos a la supervisión lo podemos encontrar en [Lunze, 1998], y un ejemplo de diagnosis aplicada a una planta química (DISARM) lo tenemos en [Vinson *et al.*, 1992] donde se proponen los modelos cualitativos para la descripción de los comportamientos de una planta química en un nivel muy abstracto, permitiendo

la diagnosis de un rango amplio de fallos con menos modelado explícito de los posibles fallos. DISARM esta basado en modelos y experiencia.

El sistema QDOCS [Subramanian and Mooney, 1996] utiliza modelos QSIM para diagnosticar fallos múltiples en sistemas dinámicos continuos. Incluye un método para propagar dependencias resolviendo un problema general de satisfacción de restricciones y un método para verificar la consistencia del comportamiento con un modelo temporal.

Modelado híbrido

Como hemos visto el modelado cualitativo soluciona en muchas ocasiones el problema de la falta de modelo, pero por el contrario se hace más impreciso en sus soluciones, debido a la falta de precisión que lleva implícito este tipo de modelado. A su vez el modelado cuantitativo es mucho más preciso, pero mucho más difícil de manejar, y necesita acudir a métodos numéricos muy complejos, en la mayoría de las ocasiones, y que necesitan una gran capacidad de cómputo.

De esta forma el modelado híbrido o semicualitativo propone quedarse con las ventajas de cada uno de esos tipos de modelado y eliminar los inconvenientes. Así pues cuando no se dispone de información cuantitativa suficiente, o esta es difícil de obtener, se recurre al modelado cualitativo, pero sin despreciar la información cuantitativa disponible.

En [Biswas *et al.*, 1999] se presenta una metodología sistemática para la diagnosis de sistemas de ingeniería complejos combinando un análisis cualitativo y cuantitativo para generar candidatos de fallos más precisos. Los candidatos iniciales son generados a partir de técnicas de satisfacción de restricciones cualitativas, y son posteriormente refinados introduciendo selectivamente información semicuantitativa derivada del modelo analítico.

En [Sainz *et al.*, 2002] podemos encontrar el uso del *análisis intervalar modal* [Group, 1999] para conseguir estimaciones de envolventes con el objeto de detectar e identificar fallos en un sistema de tres tanques interconectados. El análisis intervalar modal extiende los números reales a intervalos, y a diferencia del análisis intervalar clásico, que identifica un intervalo a partir de un conjunto de números reales, el análisis intervalar modal identifica el intervalo a partir de un conjunto de predicados que están totalmente cubiertos por los números reales.

3.3.6. Razonamiento para la diagnosis

Razonamiento basado en consistencia

La diagnosis basada en consistencia usa modelos de comportamiento correcto del sistema a diagnosticar para llevar a cabo la detección y localización de los componentes de fallo. La inclusión de modelos de modos de fallos será usada para identificar los fallos, y en ese caso se conoce como diagnóstico basado en consistencia con modos de fallo. Puesto que las ideas para la diagnosis en sistemas estáticos han sido presentada como principal exponente de la comunidad DX, extenderemos dichas ideas para la diagnosis de sistemas dinámicos.

La clave de este tipo de diagnosis se encuentra en comprobar si un conjunto de observaciones OBS contradice los modelos de ciertos modos de comportamiento del mecanismo, el comportamiento correcto (para detectar un fallo) o los comportamientos con fallos (para identificar el fallo) que resultan de ciertos componentes defectuosos y/o fallos estructurales en el mecanismo (figura 3.4). Si un modelo de comportamientos de un modo modelo (modo), y un conjunto de OBS se consideran como teorías lógicas, la tarea es comprobar la consistencia de la unión:

$$Modelo(modo) \bigcup OBS \vdash^{?} \perp$$

Si el modelo de comportamientos no captura los aspectos temporales, es un conjunto de restricciones, **restricciones-estado**, que restringe el conjunto de estados posibles bajo el modo respectivo. Se puede pensar que el modelo se representa como Estados(modo).

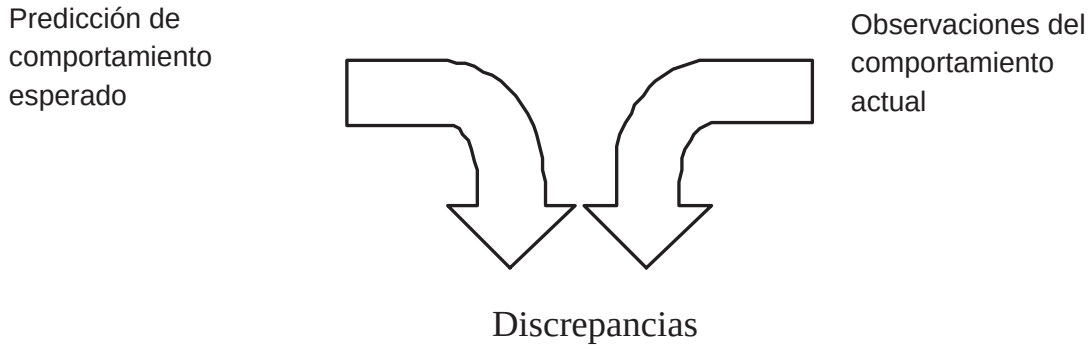

Figura 3.4: Detección de discrepancias

Pero para caracterizar la evolución en el tiempo no solamente debe haber restricciones de los estados sino restricciones de las relaciones entre estados a través del tiempo:

$$Modelo(modo) = \text{restricciones-estado(modo)} \bigcup \text{restricciones-temporales(modo)}$$

La representación de un modelo puede ser:

$$\{V, Dom(V), T\}$$

Donde V son las variables, $Dom(V)$ describe el conjunto de todos los estados posibles, y T es el universo de los instantes de tiempo. Un comportamiento es un mapeo de:

$$T \rightarrow Estados \subseteq DOM(V)$$

O como un grafo de este mapeo:

$$\{t, v(t) | t \in T\}$$

Aunque la diagnosis basada en consistencia fue la técnica pionera en la diagnosis basada en modelos, no por ello deja de evolucionar y presentar nuevas alternativas. De esta forma en [Gasca *et al.*, 2001], se presenta una aproximación para mejorar y automatizar el proceso de diagnosis en sistemas cuyos modelos se componen de restricciones de igualdad polinómica. Para ello se utilizan algoritmos de procesamiento simbólico del modelo inicial (*bases de Gröbner*), que generan un modelo reducido. Este modelo será tratado por algoritmos incrementales con el objeto de aumentar la eficiencia del proceso de diagnosis. En la misma línea [Pulido *et al.*, 2001] usa modelos cuantitativos y registros de dependencias *off-line* para reducir el cálculo de los posibles conflictos.

Razonamiento basado en restricciones

La aproximación basada en restricciones no propone una técnica de modelado particular, sino que todo el sistema se describe como un conjunto de restricciones, de tal forma que dichas restricciones pueden ser asociadas tanto a componentes como a procesos.

De manera general un *Problema de Satisfacción de Restricciones* (CSP)[4] parte de un conjunto de variables X pertenecientes a un dominio D, y un conjunto de restricciones C que limitan el conjunto de valores permitidos para esas variables.

Figura 3.5: Solución de un problema CSP

Cuando se tiene una representación de un problema CSP, podemos usar diferentes métodos para resolverlo independientemente del contexto de la aplicación. Las técnicas principales de resolución son: Búsqueda e Inferencia. Hay muchos algoritmos para realizar la búsqueda tales como: Backtracking, Branch and Bound y Programación lineal, entera y mixta.

En un modelo de satisfacción de restricciones, de un problema de diagnosis, las restricciones capturan la estructura y comportamiento del sistema a modelar, y representan las relaciones entre los atributos de los componentes del sistema.

Un atributo de un componente es modelado como una variable CSP, caracterizada por los valores posibles con los que la variable puede ser instanciada, de acuerdo a las especificaciones de diseño o las medidas de observación del atributo modelado. El sistema de diagnóstico encuentra un fallo si alguna restricción no puede ser satisfecha, indicando además, la restricción violada, la causa del fallo.

En [Sabin and Freuder, 1996] se propone la arquitectura de un sistema de diagnóstico automático para servicios de red (ADNET). ADNET construye automáticamente programas de diagnóstico especializados a partir de una librería de herramientas de pruebas de redes, librerías de herramientas de diagnosis basada en restricciones y el modelado CSP.

En [Sabin *et al.*, 1995] se propone el uso de los CSP parciales (*PCSP o Partial Constraint Satisfaction Problem*) para diagnosticar problemas de software en redes de ordenadores. Para hacer esto modela cada una de las capas de la red como un CSP y localiza el error encontrando la restricción o restricciones que no se cumplen.

[4] Constraint Satisfaction Problem

En un CSP parcial se permite que no se satisfagan algunas de las restricciones debido a que el problema está sobrerestringido o porque la solución completa requeriría demasiado tiempo de cómputo. La diagnosis minimal se corresponderá, por tanto, con la solución del PCSP que deja menos restricciones sin satisfacer.

Razonamiento probabilístico y bayesiano

Dentro de las teorías acerca del razonamiento humano, el razonamiento probabilístico es considerado un tipo de razonamiento que se apoya en los modelos de la teoría de probabilidades. El razonamiento probabilístico, provee de un mecanismo para evaluar las consecuencias de sistemas que son afectados por incertidumbres probabilísticas. Su principal característica es su habilidad para actualizar la respuesta de los sistemas condicionada a la nueva información disponible.

En [Lunze and Schiller, 1999] encontramos un ejemplo de diagnosis usando un modelo lógico probabilístico con información cualitativa. Los modelos probabilísticos no sólo describen, en el campo del diagnóstico, las relaciones causa-efecto activadas por los fallos, sino también la incertidumbre que ocurre dentro de dichas relaciones.

Por su parte las redes de Bayes proporcionan una forma de computar a posteriori probabilidades sobre las hipótesis (que permiten fácil discriminación). Este tipo de aproximaciones es usada frecuentemente en medicina, pero muy usada también en la generación de sistemas expertos.

Una red de Bayes es un grafo acíclico dirigido que proporciona una representación para la información de independencia condicional. Cada variable se corresponde con un nodo en el grafo. La distribución condicional de un nodo dados sus padres se guarda dentro del nodo.

La semántica de independencia de una red de Bayes proporciona un esquema legítimo para calcular independencias condicionales inferidas a partir de las declaradas explícitamente (por ejemplo por la estructura de la red). Cuando construimos una red de Bayes para un dominio, el orden de las variables se selecciona siguiendo la dirección 'causal', es decir, los antecesores de un nodo son las causas y los descendientes los efectos. Esta es la forma más natural de elegir el orden para modelar el dominio.

Se han aplicado las redes de Bayes a la construcción de sistemas expertos de diagnosis en dominios donde es crucial un manejo coherente de la incertidumbre sobre el diagnóstico. Uno de los mayores factores de motivación para esta aplicación es que la extensión de los sistemas basados en reglas al ocuparse de la incertidumbre, como los factores de certeza, no puede dar una semántica clara a menos que se impongan limitaciones muy fuertes en la estructura de la base de conocimiento [Horvitz and Heckerman, 1986].

De la misma forma que los sistemas expertos basados en reglas, la mayoría de los sistemas de diagnóstico basado en redes de Bayes fueron construidos a mano por los expertos. En el campo de la diagnosis médica se han construido varias aplicaciones significativas, por ejemplo [Heckerman *et al.*, 1992]. En [Srinivas, 1994; Darwiche, 1995] se usa la sinergia entre las técnicas de redes de Bayes y los métodos usados en diagnosis basada en modelos probabilísticos .

Razonamiento basado en abducción

La formalización de diagnosis usando modelos de dominio causal se conoce generalmente como diagnosis abductiva. Como típico ejemplo de este tipo de diagnosis tenemos a [Console *et al.*, 1989; Console and Torasso, 1991] en cuyas teorías de *diagnosis abductiva* el comportamiento normal o anormal de un sistema es modelado en términos de conocimiento causal con estados normales o anormales y resultados predichos. En esta teoría se distinguen dos tipos de conocimiento causal. En el primer tipo de conocimiento causal se asume que, cuando está presente un conjunto de fallos, deben estar presentes también todos los resultados asociados. A esta noción de causalidad se le llama *causalidad fuerte*. El segundo tipo de conocimiento causal, los resultados relacionados con la causalidad no necesariamente deben estar presentes cuando se presentan sus fallos asociados. Esta noción menos estricta de causalidad se conoce como *causalidad débil*, y representa las relaciones de dudosa imprecisión entre la causa y el efecto.

Las relaciones de causalidad fuerte entre fallos, y entre fallos y resultados observables se denotan en la teoría de la diagnosis abductiva por una implicación lógica de la forma:

$$d_1 \wedge .... \wedge d_n \rightarrow d$$

y

$$d_1 \wedge .... \wedge d_n \rightarrow f$$

que expresan que la ocurrencia combinada de los fallos $d_1, ..., d_n$ provoca el fallo d y el resultado f. Los fallos d y los resultados f son representados como predicados lógicos.

Una especificación causal $C = (\Delta, \Phi, R)$ se define como un conjunto de literales de fallos Δ, literales de resultados Φ y una colección de implicaciones lógicas R relacionando literales de fallos y de resultados de la forma que se ha visto anteriormente. Las implicaciones lógicas R son conocidas normalmente como *axiomas de anormalidad*, debido a que representan sólo el conocimiento causal de la anormalidad.

Sea $A = (C, E)$ un problema de diagnosis abductiva, donde C son implicaciones causales y E resultados observados, entonces el conjunto de fallos $H \subseteq \Delta$ se dice que es una diagnosis de A si y sólo si:

1. $\forall f \in E : R \cup H \vDash f$ (*Condición de cobertura*) y
2. $\forall f \in E^c : R \cup H \nvDash \neg f$ (*Condición de consistencia*)

donde E^c, el conjunto de resultados observables se asume que está ausente, se define en términos de E como sigue:

$$E^c = \{\neg f \in \Phi | f \notin E, f \textit{ es un literal positivo } \}$$

Esto significa que una diagnosis H debe predecir todos los resultados observados, pero no puede predecir los resultados que se asume que están ausentes.

El correspondiente concepto de diagnosis se llama el concepto de **diagnosis de causalidad fuerte (SC)** y se define como sigue:

$$SC_{\Sigma,\varrho|H}(E) = \begin{cases} H \text{ si } {}_{\varrho|H}(H) = E \\ \text{en otro caso } u \end{cases}$$

El concepto diagnosis que corresponde a la diagnosis abductiva con relaciones de causalidad débil se conoce como el concepto de **diagnosis de causalidad débil** y se denota por **WC**. Se define como sigue:

$$WC_{\Sigma,\varrho|H}(E) = \begin{cases} H \text{ si } {}_{\varrho|H}(H) \supseteq E \\ \text{en otro caso } u \end{cases}$$

La diagnosis de causalidad débil considera todos los resultados observados, aunque no todos los resultados observables necesitan ser observados.

La diagnosis abductiva ha sido utilizada en varios desarrollos como pueden ser [Poole *et al.*, 1987; Konolige, 1994; Poole, 1994].

3.4. Diagnosis mediante Aprendizaje Automático

Los métodos de aprendizaje y reconocimiento de patrones son muy útiles cuando se pretenden diagnosticar sistemas donde no se dispone del conocimiento estructural para relacionar los síntomas con los fallos. Desde este punto de vista, la tarea de diagnosis se entiende como la capacidad de diferenciar entre n_f fallos diferentes $F_j, j \in \{1, ..., n_f\}$, usando n_s síntomas $s_i, i \in \{1, ..., n_s\}$.

Los fallos son representados mediante un vector de fallos $F = [F_1 F_2 .... F_n]$ y los síntomas mediante un vector de síntomas $s = [s_1 s_2 .... s_{ns}]^T$

Los métodos de aprendizaje determinan experimentalmente mediante el entrenamiento previo, el fallo F_j en base al vector de síntomas de referencia s_{ref}. La comparación posterior con los síntomas observados s, determinará el fallo mediante la clasificación obtenida, tal como se muestra en la figura 3.6

Con estas técnicas, la diagnosis se consigue mediante la comparación de la evolución del sistema a diagnosticar con el conocimiento aprendido que se tiene del mismo, evaluando de esta forma el comportamiento del sistema. Para la adquisición del conocimiento se suele recurrir a experiencias previas almacenadas o a la simulación del comportamiento del sistema a diagnosticar. Dicho conocimiento puede dar lugar a un modelo explícito o no. Un ejemplo del primer caso es el aprendizaje simbólico. Como representación del segundo tenemos las técnicas con Redes Neuronales Artificiales.

Una de las grandes ventajas de las técnicas de Aprendizaje Automático es que evitan los problemas relacionados con el desarrollo de un modelo explícito de conocimiento o la obtención de algún tipo de modelo, requeridos por las aproximaciones basadas en conocimiento o basadas en casos, respectivamente. Por el contrario su gran inconveniente radica en la necesidad de un conjunto de datos de entrenamiento, normalmente extenso, que no suele estar disponible en los sistemas reales, por lo que es necesario acudir, con excesiva frecuencia, a simulaciones de los

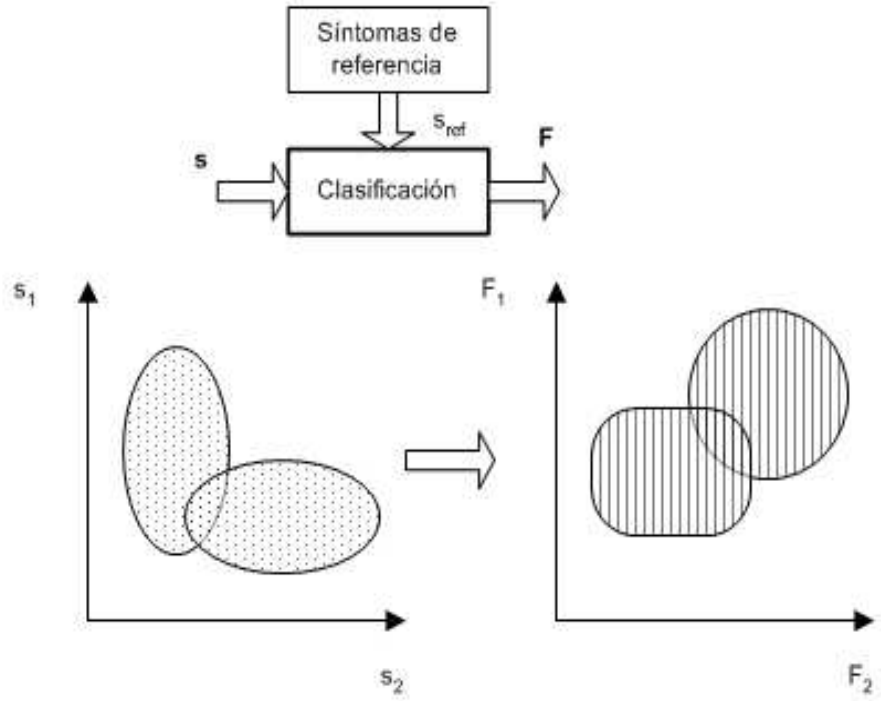

Figura 3.6: Diagnosis mediante clasificación automática

mismos [Balakrishnan and Jonavar, 1998]. Otra de las desventajas que presentan estos sistemas aplicados a la diagnosis es la imposibilidad de reconocer situaciones para los que no han sido entrenados.

A continuación se revisarán las técnicas más habitualmente usadas para realizar diagnosis mediante el uso de aprendizaje automático.

3.4.1. Redes Neuronales

Las redes neuronales realizan un mapeado no lineal entre las entradas y las salidas, consistente en interconectar neuronas distribuidas en capas. Las capas se conectan de tal forma, que las señales de las capas de entrada se propagan a través de la red. El comportamiento no lineal de una red neuronal viene determinado por la elección en la topología de la red, y el peso de conexión entre las neuronas.

Las redes neuronales fueron los primeros sistemas de aprendizaje aplicados a la diagnosis, y su campo de aplicación es muy amplio, así como el tipo de red empleada para llevarla a cabo. De esta manera en [Pacheco *et al.*, 2001] podemos

encontrar una red de tipo *perceptrón multicapa* para llevar a cabo la detección de fallos en los robots de soldadura de una factoría de automóviles o en [Saludes *et al.*, 2001] se usa una *red de Kohonen* o *red Self Organising Map (SOM)* para la detección de fallos en un grupo hidroeléctrico.

En [Lennox *et al.*, 1998] se aplican las redes neuronales para capturar las características no lineales del proceso de vitrificación que evita la perdida de líquido y proporciona un medio de almacenamiento seguro. El modelo obtenido por la red neuronal es capaz de detectar también el fallo inminente del recipiente usado en el mismo proceso de vitrificación.

En [Fuente, 2001] se exponen tres formas distintas de aplicar las Redes Neuronales Artificiales al campo de la detección y diagnóstico de fallos. En primer lugar la red neuronal es empleada para identificar un modelo no lineal de una planta a partir de los datos de entrada-salida, usando posteriormente dicho modelo para la generación de residuos y la matriz de incidencias, usando de esta forma técnicas tradicionales de la comunidad FDI para llevar a cabo la diagnosis. En segundo lugar las redes neuronales son usadas como clasificador de información, de tal forma que a partir de los datos medidos del proceso se entrena la red para clasificar y diferenciar patrones de operación normales de la planta de patrones de operación con fallo. Para ello usa un ejemplo sobre un reactor químico donde induce los fallos que posteriormente se quieren diagnosticar para que la red neuronal aprenda a clasificarlos. La tercera aplicación que realiza de las redes neuronales en el campo de la detección y diagnóstico de fallos es estudiar el contenido en frecuencias de una señal representativa del proceso, que tiene información tanto de los fallos como de las perturbaciones y el ruido que afectan al proceso. Así pues la red debe aprender a clasificar patrones de fallos de los patrones de perturbación e intentar encontrar un modelo robusto.

En [Samanta and Al-Balushi, 2003] se presenta una red neuronal para diagnosticar fallos en los cojinetes de elementos rodantes. Para ello se utiliza como entrada de la red la señal de vibración a través del tiempo, tanto para rodamientos defectuosos como válidos. En [Yang *et al.*, 2004] se usa una red neuronal para la diagnosis de fallos en motores eléctricos. En este caso se combinan las redes ART con las de Kohonen para conseguir el diagnóstico. Sin embargo, en [Kowalski and Orlowska-Kowalska, 2003] se combina el uso de un perceptrón multicapa con

una red SOM para alcanzar el diagnóstico en motores de inducción para fallos del estator, el rotor y los rodamientos.

En [Vemuri and Polycarpou, 2004] se plantea el uso de una red neuronal para diagnosticar fallos en robots manipuladores y en [Power and Bahri, 2004] se describe un marco de diagnóstico y supervisión de fallos usando redes neuronales. Dicho marco es usado para implementar un sistema de detección de fallos que es capaz de determinar la localización exacta de los fallos y diagnosticarlos en una planta piloto.

Esto son tan sólo algunos ejemplos de la diversidad de redes neuronales utilizadas en los muy diversos ámbitos del diagnóstico automático. Un análisis exhaustivo revelaría cientos de trabajos y aplicaciones que usan las redes neuronales para el diagnóstico, por sí mismas o en conjunción de otras técnicas.

3.4.2. Árboles de decisión

Los árboles de decisión son modelos de clasificación secuencial que se construyen mediante la partición de la muestra inicial en varios subconjuntos descendientes a través de algún test lógico, conectándose los distintos nodos mediante ramas. El proceso de partición continúa hasta que se cumple una determinada condición de parada, dando lugar a un conjunto de nodos terminales (hojas). Además aprenden modelos sobre una única tabla de datos y que no establecen relaciones entre más de una fila de la tabla a la vez ni sobre más de un atributo a la vez. A los modelos así generados se les conoce como *modelos proposicionales* [Hernández Orallo *et al.*, 2004].

En [Cascio *et al.*, 1999] se presenta un caso de aplicación de aprendizaje supervisado usando árboles de decisión aplicado a la diagnosis a bordo del sistema de inyección de combustible *Common-Rail* de un automóvil. La justificación expuesta para el uso de los árboles de decisión se fundamenta en que para la diagnosis *on-board* es mucho más eficiente usar este tipo de tecnologías, que pueden ser fácilmente implementadas con pocas necesidades de memoria y cuyo tiempo de computación es muy bajo.

En [Chen *et al.*, 2004] se usan los árboles de decisión para diagnosticar fallos en servidores de internet. Se recogen las propiedades de cada petición y se generan los árboles de decisión en base a los datos recogidos para localizar los fallos. El estudio

presentado es evaluado comprobando peticiones del servidor eBay, demostrando la viabilidad de la técnica.

En [Lee, 2005] se utilizan los árboles de decisión para diagnosticar el estado de un motor de inducción partiendo de las señales de vibración del motor. Estas señales de vibración son también la base para la aplicación de técnicas de aprendizaje en el diagnóstico de fallos en turbinas de vapor en [Zhao *et al.*, 2005]

Los árboles de decisión han sido también ampliamente usados en la diagnosis médica, donde la cantidad de datos disponibles hace que las técnicas de aprendizaje sean especialmente atractivas. Así en [Vlahou *et al.*, 2003] se usa el algoritmo de clasificación comercial *biomarker patterns software* (BPS), que está basado en CART[5] para diagnosticar el cáncer de ovario, y en [Yahui *et al.*, 2007] se usan para la diagnosis del cáncer gástrico.

3.4.3. Máquinas de Soporte Vectorial

Las máquinas de soporte vectorial [Vapnik, 1998] han demostrado ser una herramienta muy efectiva cuando se aplica a problemas de reconocimiento de patrones. La idea fundamental es encontrar un hiperplano que separe las clases del conjunto de entrenamiento. Para ello, aumentan la dimensionalidad del conjunto de entrada mediante una función núcleo que permite encontrar un discriminante lineal en el espacio dimensional ampliado.

En los últimos años, han venido siendo ampliamente usadas en diferentes ámbitos, debido a su robustez y fiabilidad.

En [He and Shi, 2002] se demuestra que las máquinas de soporte vectorial producen resultados más precisos que las redes neuronales cuando son aplicados a problemas de diagnóstico de fallos en válvulas de bombas recíprocas usando datos de vibración. Se usan *transformaciones wavelet* para preprocesar los datos de vibración, extraer información temporal y de frecuencia, y entonces se usan las SVM para clasificar los fallos. En [Ge *et al.*, 2004] también se compara el uso de las SVM con las redes neuronales. En este caso, la SVM se usa para la diagnosis de fallos en las operaciones de estampado de las planchas de metal. Demuestra mejores resultados, incluso con un conjunto de entrenamiento menor.

[5] un conocido algoritmo para la construcción de árboles de decisión [Breiman *et al.*, 1984]

En [Saunders and Gammerman, 2000] se aplican las SVM a la diagnosis y reparación automática. Para ello utiliza los registros de logs de los entornos de manufacturación. Estos registros se usan para diversas tareas, como asistencia al control de stock y monitorización y mejora de los procesos productivos. Estos registros históricos son usados en este trabajo para reconocimiento de patrones que permitan determinar fallos y ayudar en la reparación.

En [Ge *et al.*, 2004] se usan las SVM para diagnosticar los fallos en las soldaduras de planchas de metal. Los resultados obtenidos son comparados con los conseguidos por una red neuronal, mostrándose más efectiva la SVM incluso con menor número de casos de entrenamiento.

En [Yuan and Chu, 2006] se usan las SVM para diagnosticar los fallos del rotor de una turbo-bomba. En este trabajo, se presenta un método para clasificar múltiples clases usando una clasificación uno contra todos y construyendo un árbol de clasificación binario compuesto por múltiples clasificadores binarios.

3.4.4. Otras técnicas

Además de las técnicas de aprendizaje anteriormente mencionadas, existen otras técnicas de aprendizaje que, aunque en menor medida, también son aplicadas a la diagnosis. Además, se han publicado multitud de trabajos donde no sólo se recurre a una técnica concreta, sino a la combinación de varias de ellas. A modo de ejemplo, podemos citar algunos de estos trabajos.

En [Simón *et al.*, 2001] podemos encontrar técnicas conexionistas combinadas con lógica borrosa y sistemas basados en conocimiento para llevar a cabo la diagnosis del glaucoma. Esta técnica transforma los parámetros que usa el oftalmólogo para llevar a cabo su diagnóstico en conjuntos borrosos, y recoge la información de uno de ellos, el campo visual, a partir de un sistema de clasificación basada en redes neuronales artificiales. Todo este conocimiento se representa finalmente mediante hechos y reglas de producción.

En [Aguado and Aguilar, 1999] encontramos un método de clasificación de autoaprendizaje, que trabaja con información cuantitativa y cualitativa que puede funcionar tanto en modo supervisado como no supervisado, aplicado a dos ejemplos, uno sobre datos reales de una planta piloto de dos tanques de agua interconectados, y el otro sobre un modelo que simula el comportamiento de una red

de comunicaciones. Este algoritmo está basado en el poder de generalización de la lógica borrosa [Zadeh, 1965] y la capacidad de interpolación de las conectivas lógicas híbridas [Piera, 1987]. En [S.O.T. Ogaji and Prober, 2005] se usa también la lógica borrosa para diagnosticar fallos en turbinas de gas.

Los sistemas neuro-borrosos representan un conjunto de reglas borrosas puestas bajo la forma de una red neuronal con el objetivo de hacer uso del aprendizaje, adaptación y posibilidades de paralelismo ofrecidas por las redes neuronales. En [Calado *et al.*, 2001] se presenta una propuesta de uso de los sistemas neuro-borrosos para la diagnosis de sistemas dinámicos.

Las técnicas de combinación de clasificadores como el *Boosting*, han sido también aplicadas al campo de la diagnosis. Por ejemplo, en [Rodriguez and Alonso, 2002] se hace un estudio de la viabilidad de dichas técnicas.

Para diagnosticar una planta generadora de potencia, en [Garcez *et al.*, 1997] se usa una extensión de C-IL2P que permite la implementación de programas lógicos en redes neuronales. Dicha extensión, posibilita que C-IL2P sea aplicable a problemas donde el conocimiento de fondo se representa en forma lógica.

3.5. Problemas abiertos

A pesar de la diversidad de técnicas utilizadas y los diferentes enfoques que se aplican al campo de la diagnosis, existen aún una serie de problemas que no están totalmente resueltos o presentan un amplio campo de discusión. Entre ellos destacaremos los siguientes:

- La capacidad de aplicación de estas técnicas a sistemas más grandes o más complejos. Aunque muchas de las técnicas presentadas son novedosas y solucionan elegantemente el problema que plantean, no es menos cierto que en bastantes ocasiones dichas técnicas sólo han sido probadas con modelos o en pequeñas aplicaciones, sin tener en cuenta el problema que conllevaría trasladar dichas aproximaciones a sistemas más grandes y complejos, y sobre todo a problemas reales e industriales.

- El modelado del sistema. En cuanto al enfoque más prolíficamente usado en el campo de la diagnosis, la Diagnosis Basada en Modelos, todas las propuestas se basan en la existencia de un modelo previo, pero nadie se plantea el

problema del modelado del sistema como una de las tareas más importantes y fundamentales en este tipo de aplicaciones. De hecho deberíamos seleccionar el modelo de acuerdo al costo computacional y exactitud requeridos según el tipo de sistema a diagnosticar.

- La dinámica de los fallos. Los fallos pueden cambiar dinámicamente, y nosotros podemos encarar el problema de modelar fallos intermitentes o aprovecharnos de la suposición de que no existen fallos intermitentes. Un fallo intermitente puede causar también una desviación permanente en el comportamiento del sistema donde un fallo permanente crearía síntomas sólo ocasionalmente. Por otra parte asumir la monotonía de un fallo incipiente supone una simplificación que no siempre nos podemos permitir.

- Diagnosis en tiempo real. El comportamiento cambiante en el tiempo de los sistemas, en conjunción con los requisitos para prevenir o minimizar el daño y el coste, pueden imponer restricciones sobre el tiempo disponible para realizar la diagnosis. Se necesita pues una arquitectura práctica de tiempo real que se guíe por el objetivo de tener una diagnosis útil (potencialmente refinable) disponible en cualquier momento.

- Diagnosis y reparación. Los procesos reales de diagnóstico incluyen razonar y actuar así como la idea de restablecer el funcionamiento deseado. Pero todo lo que hemos venido discutiendo se centra en la idea de encontrar el fallo, no de repararlo. El hecho de que la diagnosis debería servir como tarea reparadora debe tener un impacto en el sistema de diagnóstico. Por ejemplo, no tiene ningún sentido gastar energías en distinguir dos fallos que necesitan la misma reparación. Las acciones de reparación podrían apoyar la localización del fallo (por ejemplo reemplazando un componente dañado o punteándolo mediante un cambio estructural), y necesitaríamos teorías y sistemas que integrasen ambas tareas. En este sentido, la automatización (*autonomic computing*) es un tema candente actualmente. La tecnología que automáticamente detecte y remedie los problemas es una tentadora perspectiva.

- Fallos estructurales. El concepto de diagnosis que hemos venido usando se centra en la conservación de la estructura original del sistema que se quiere

diagnosticar. A través de la definición de diagnosis como un modo de asignación, los fallos se limitan a componentes 'rotos' y se asume que la estructura del sistema siempre permanece intacta. Pero el fallo puede ser precisamente una violación de dicha estructura, por ejemplo, que se rompa una de las conexiones de un circuito electrónico, o un calentamiento imprevisto de un componente por una fuente de calor. Una solución para las conexiones rotas puede ser modelar dichas conexiones como componentes del sistema que también pueden fallar, lo cual incrementa la complejidad, pero no servirían para conexiones accidentales de componentes o interacciones imprevistas entre ellos debido a múltiples causas.

- Fallos aditivos o múltiples y fallos compensados. Muchos de los trabajos asumen como punto de partida que sólo va a ocurrir un fallo en el sistema al mismo tiempo. Si eliminamos esta simplificación la complejidad se dispara a la hora de tener que diagnosticar la posibilidad de detectar más de un fallo al mismo tiempo, debido a que la combinación de dos o más fallos puede provocar que el conjunto de posibilidades a evaluar crezca en exceso. Además en estos casos siempre cabe la posibilidad de que un determinado comportamiento del sistema se pueda corresponder con distintas combinaciones de fallo. Bajo estas características un problema adicional ocurre cuando se producen fallos compensados, que son aquellos en los cuales un fallo tapa la ocurrencia del otro, por ejemplo una obstrucción en una conducción y una fuga de la misma magnitud.

- Diagnosis Distribuida. Uno de los campos que están siendo estudiados más profusamente en los últimos años es la posibilidad de realizar diagnosis de forma distribuida. Existen dos propuestas principales para la realización de diagnosis distribuida de sistemas complejos:

  Una primera propuesta consiste en definir una partición de la estructura del sistema y asignar un agente a cada región de la partición. Cada agente realiza diagnosis local de la partición asignada. La diagnosis global se obtiene definiendo un método apropiado de comunicación entre los agentes [Wörn *et al.*, 2004]

La otra posible propuesta es reunir la especialización del diagnóstico de diferentes metodologías [Köpen-Seliger *et al.*, 2003]. Es decir, los agentes representan, en este caso, diferentes metodologías de diagnosis. La tarea principal del sistema de diagnóstico es decidir, para cada región de la partición, las metodologías de diagnóstico disponibles (agentes) que proporcionan los mejores resultados.

Las dos implementaciones carecen de una metodología de particionado coherente de la estructura del sistema monitorizado en un conjunto de regiones, tal que el nivel de independencia del proceso de diagnosis local para cada región es máxima y la comunicación entre las diferentes regiones, necesaria para la diagnosis global, es mínima.

La metodología de particionado propuesta en [Bocaniala and da Costa, 2005] divide el sistema monitorizado en regiones totalmente independientes, es decir, máximo nivel de independencia del proceso de diagnostico local. Los agentes usados en esta propuesta tienen independencia causal completa debido a la independencia causal de la región correspondiente en la partición. Esto permite realizar la diagnosis en cada región de forma local, sin necesidad de comunicación con el resto de regiones. Esto permite, además, mantener la atención de la diagnosis exclusivamente en aquellas que están afectadas por el fallo.

Capítulo 4

Extracción de Conocimiento a Partir de Bases de Datos

4.1. Introducción

En los últimos años se ha apreciado un enorme crecimiento en la capacidad de generación y almacenamiento de información. Entre otros, podemos citar tres factores decisivos en la contribución a este crecimiento:

- Los avances de la tecnología, que ofrecen almacenar y procesar grandes cantidades de datos a un precio relativamente bajo.
- La velocidad con la que se almacenan dichos datos.
- El interés científico y comercial por explotar los grandes volúmenes de información almacenada.

Desafortunadamente, no se ha producido un desarrollo equivalente en las técnicas de análisis de información, por lo que existe la necesidad de crear una nueva generación de técnicas y herramientas informáticas con capacidad para ayudar a los usuarios finales en el análisis automático de los datos.

Se puede decir que el principal objetivo de estas técnicas es el de procesar automáticamente grandes cantidades de datos para encontrar conocimiento útil para los usuarios.

Este proceso, más conocido con las siglas *KDD* (*Knowledge Discovery Databases*), fue definido por Frawley como:

"la extracción no trivial de información implícita, previamente desconocida y potencialmente útil a partir de los datos" [Frawley *et al.*, 1992]

Esta definición fue revisada por Fayyad en el año 1996 como:

.el proceso no trivial para identificar patrones válidos, potencialmente útiles y en última instancia comprensibles a partir de los datos" [Fayyad *et al.*, 1996]

De una manera más informal podemos decir que el $\mathcal{KDD}$ es el proceso no trivial de extracción de información oculta, desconocida, válida y útil a partir de bases de datos. La información obtenida debe ser oculta y desconocida, esto es, no debería ser un conocimiento deducible desde los datos puesto que, en cierta medida, es una información no esperada por el usuario. También debe ser válida para que los patrones descubiertos puedan aplicarse en el futuro, y por último, la información descubierta ha de ser útil y entendible por el usuario, es decir, debe ser presentada como conocimiento apropiado para satisfacer las metas del usuario y que, además, pueda interpretar fácilmente los resultados.

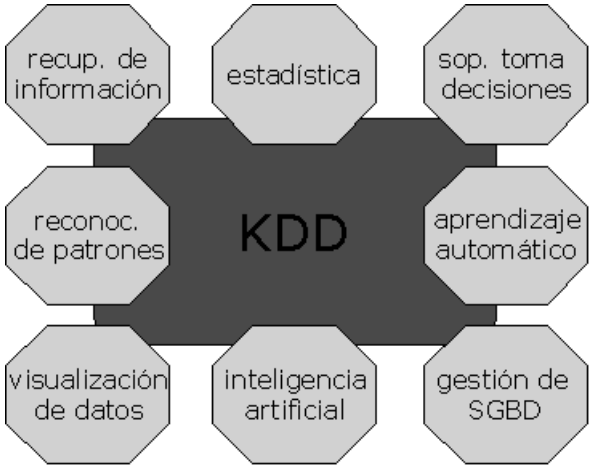

Figura 4.1: Interdisciplinariedad en el proceso de $\mathcal{KDD}$

El $\mathcal{KDD}$, como muestra la figura 4.1, es un campo interdisciplinario que involucra distintas áreas de investigación tales como la estadística, el soporte a la toma de decisiones, el aprendizaje automático, la gestión y almacenamiento de bases de datos, la inteligencia artificial, las técnicas de visualización de datos, el reconocimiento de patrones o la recuperación de la información. Para la realización de estos procesos se aplican técnicas procedentes de muy diversas áreas, como

pueden ser los algoritmos genéticos, las redes neuronales, los árboles de decisión, la lógica difusa, el razonamiento probabilístico, etc.

4.2. Fases del proceso del *KDD*

Aunque a veces el proceso de *KDD* se simplifica únicamente a la fase de Minería de Datos (*Data Mining*), el proceso completo, como se puede apreciar en la figura 4.2, consta de una serie de fases o etapas hasta alcanzar los objetivos:

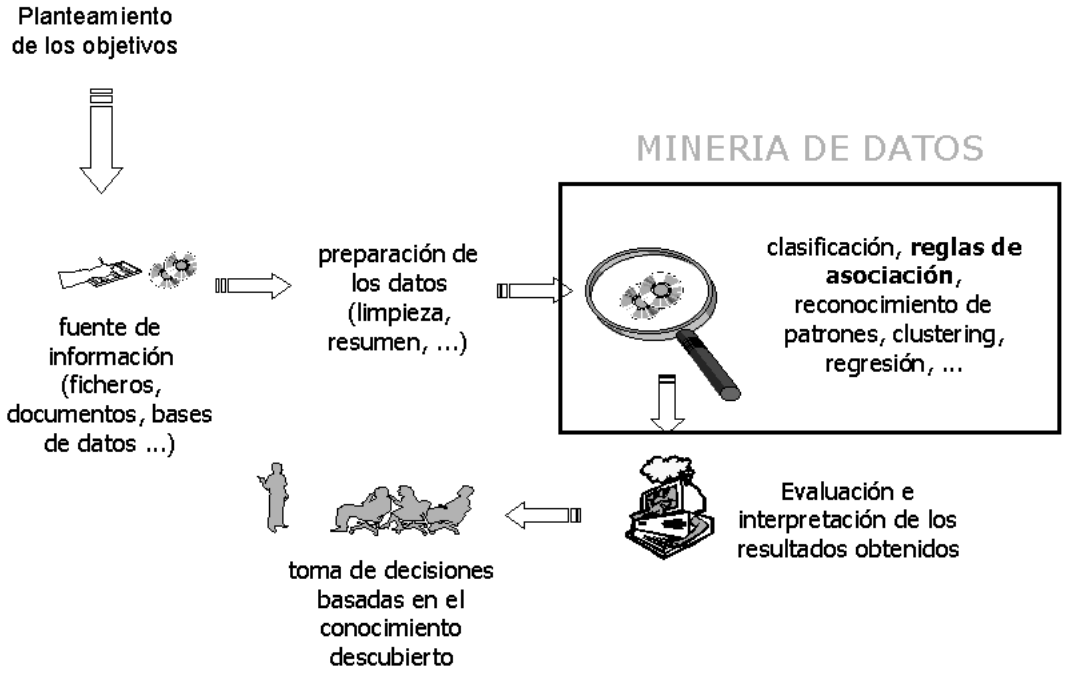

Figura 4.2: Fases del proceso de *KDD*

1. **Entendimiento del dominio de la aplicación.** Es necesario conocer los objetivos y las metas del usuario, así como las posibilidades de automatizar el proceso.

2. **Selección.** En esta fase se establecen cuáles son las distintas fuentes de información de las que se dispone y que son significativas para poder extraer conocimiento.

3. **Limpieza y preprocesado de los datos.** Es necesario homogeneizar los datos para que puedan aplicarse las técnicas de extracción de conocimiento,

diseñando estrategias para evitar ruido, valores incompletos, casos extremos (outliers), etc.

4. **Transformación y reducción de los datos.** Incluye la búsqueda de características útiles de los datos según sea el objetivo final, la reducción del número de variables y la proyección de los datos sobre espacios de búsqueda en los que sea más fácil encontrar una solución. Este es un paso crítico dentro del proceso global, que requiere un buen conocimiento del problema y una buena intuición, y que, con frecuencia, marca la diferencia entre el éxito o el fracaso del proceso de $\mathcal{KDD}$.

5. **Selección del tipo de tarea y del conocimiento a obtener.** Existen distintas estrategias para extraer conocimiento, dependiendo del objetivo final del usuario: clasificación, asociación, agrupamiento, predicción de comportamientos, detección de desviaciones, etc.

6. **Selección del algoritmo de minería de datos.** Este paso es muy importante pues de él depende la obtención de resultados aceptables.

7. **Análisis e interpretación del conocimiento extraído**. Este análisis puede inducir la repetición del proceso, quizás con otros datos, otros algoritmos, otros objetivos y otras estrategias. La obtención de resultados aceptables está influido por la existencia de medidas de interés del conocimiento (de tipo estadístico, en función de su sencillez, etc.), que permiten presentar únicamente el conocimiento útil. Además, la interpretación de estos resultados puede beneficiarse de procesos de visualización.

8. **Difusión y uso del nuevo conocimiento**. El conocimiento se obtiene para realizar acciones, ya sea incorporándolo dentro de un sistema o simplemente para documentarlo y distribuirlo a las personas interesadas.

En las siguientes secciones se analiza con mayor detalle cada una de estas etapas, ofreciendo los métodos que normalmente se utilizan en cada una de ellas.

4.3. Las fuentes de información

Las primeras fases del $\mathcal{KDD}$ determinan que las fases sucesivas sean capaces de extraer conocimiento válido y útil a partir de la información original.

En principio, los algoritmos de minería de datos no especifican cuál debe ser el medio o soporte de los datos. Lo ideal es que el algoritmo sea capaz de aplicarse a cualquier tipo de almacenamiento de información. Puesto que actualmente existen varios sistemas de almacenamiento, la minería de datos, sobre todo a nivel comercial, está dedicando gran parte de su esfuerzo en la fase de selección de los datos para que sean compatibles con la mayoría de los tipos de almacenamiento de información. A continuación se detallan algunos de los más habituales:

- **Ficheros planos.** Actualmente es la fuente de información más común para los algoritmos de minería de datos. Normalmente son ficheros de texto o binarios con una estructura conocida por el algoritmo.

- **Bases de datos relacionales.** Quizás sea el sistema de almacenamiento más extendido. Una base de datos relacional está formada por un conjunto de tablas bidimensionales relacionadas entre sí. Cada tabla, a su vez, está formada por filas y columnas, donde las filas representan objetos o relaciones entre objetos y las columnas muestran las características o atributos de dichos objetos. Los algoritmos de minería de datos que trabajan con bases de datos relacionales son más versátiles puesto que pueden utilizar el lenguaje SQL para la fase de limpieza, preprocesado, transformación y reducción de los datos.

- **Data Warehouses.** El Data Warehouse es una especie de punto focal que guarda en un único lugar toda la información útil proveniente de sistemas de producción y fuentes externas. Antes de cargarse en el Data Warehouse, la información debe extraerse, depurarse y prepararse, y una vez integrada, la información debe presentarse de manera comprensible para el usuario.

  Una definición de Data Warehouse, propuesta en [Inmon, 1992], podría ser la siguiente: *"colección de datos orientados al sistema, integrados, no volátiles e historiados, organizados para el apoyo de un proceso de ayuda a la decisión"*.

El objetivo es lograr recomponer los datos disponibles para obtener una visión integrada y transversal a las distintas funciones de la empresa, una visión de negocio a través de distintos ejes de análisis y la visión agregada o detallada, adaptada a las necesidades.

- **Bases de datos de transacciones.** Son conjuntos de transacciones, generalmente representadas por el momento de realización, un identificador y un conjunto de objetos o items. En este caso, los registros pueden ser de tamaño variable, dependiendo del número de items que contengan. Es el tipo de almacenamiento con el que trabajan, generalmente, los algoritmos que extraen reglas de asociación.

- **Bases de datos multimedia.** Este tipo de bases de datos incluyen objetos multimedia como vídeo, imágenes y audio.

- **Bases de datos espaciales.** Son bases de datos en las que, además de la información habitual, se almacena información geográfica como mapas, coordenadas regionales y globales, etc.

- **Bases de datos temporales.** Una base de datos temporal es aquella que puede contener datos históricos y datos actuales. Los datos temporales son una forma de representar hechos relacionados con el tiempo.

- **World Wide Web.** La Web es el sistema de almacenamiento de información más heterogéneo y dinámico que existe actualmente. De hecho, una disciplina en pleno auge es el *web mining*, que se centra en la extracción de conocimiento a partir de la información generada en la web.

A continuación, se presentan una serie de definiciones relacionadas con los datos que permitirán aunar los conceptos que se utilizarán a lo largo de esta memoria de tesis.

Definición 4.1. **Dominio**. Un dominio $\mathcal{D}$ es un conjunto de valores del mismo tipo. Los dominios se pueden clasificar en cuantitativos y categóricos. Los dominios cuantitativos son aquellos cuyos valores son numéricos y pueden ser, a su vez, discretos (valores enteros) o continuos (valores reales). Los dominios categóricos

son aquellos cuyos valores no son numéricos y pueden ser nominales (sin un orden en la secuencia) u ordinales (con orden en su secuencia).

Definición 4.2. **Atributo.** Un atributo $\mathcal{A}$ es una característica de un ejemplo o caso que toma sus valores de un determinado dominio $\mathcal{D}$.

Definición 4.3. **Clase.** Una clase es un atributo especial, utilizado en aprendizaje supervisado, que sirve para categorizar o etiquetar cada uno de los casos con un valor perteneciente a su dominio. En aprendizaje supervisado, el dominio de la clase suele ser discreto.

Definición 4.4. **Ejemplo.** Un ejemplo, también denominado caso o instancia, es cada uno de los elementos pertenecientes a un conjunto de datos $\mathcal{C}$. De esta forma, un ejemplo e es un registro de la base de datos que toma sus valores de los dominios de los atributos de $\mathcal{C}$.

Definición 4.5. **Conjunto de datos.** Un conjunto de datos $\mathcal{C}$ es un subconjunto del producto cartesiano de los dominios de los atributos pertenecientes a dicho conjunto. Esto es, $\mathcal{C} \subseteq (\mathcal{D}_1 \times \mathcal{D}_2 \times ... \times \mathcal{D}_m)$, donde $\mathcal{D}_i$ es el dominio del atributo $\mathcal{A}_i$. Un conjunto de datos se caracteriza por el número m de atributos y el número de ejemplos o casos que contiene.

Definición 4.6. **Conjunto de entrenamiento.** Un conjunto de entrenamiento es un subconjunto de un conjunto de datos que será utilizado como entrada en el algoritmo de aprendizaje automático. Normalmente se utiliza sólo una parte del conjunto de datos para obtener el conocimiento.

Definición 4.7. **Conjunto de prueba o test.** Un conjunto de test es un subconjunto de un conjunto de datos que será utilizado para la evaluación del conocimiento inducido, como medida para determinar la fiabilidad del algoritmo empleado.

4.4. Limpieza, preprocesado y transformación de los datos

Una vez seleccionadas las fuentes de información, éstas deben someterse a un proceso de limpieza y transformación en el que los datos son preparados, mediante diferentes técnicas, con el objeto de obtener nuevos conjuntos de datos a los que se puedan aplicar con garantía los algoritmos de minería de datos.

Principalmente consiste en eliminar los datos erróneos o inconsistentes y los datos irrelevantes, aplicándoles el formato adecuado. De esta forma, durante estas etapas, los datos seleccionados se someten a tres tipos de técnicas: tratamiento de valores ausentes, eliminación de ruido y normalización de los datos.

4.4.1. Tratamiento de valores ausentes

La inserción de datos en una base de datos es un proceso que puede realizarse de forma automática o manualmente. En esta introducción de la información es muy posible que, en el momento del proceso, se desconozcan algunos de los valores o no se recojan correctamente. En estas situaciones se suele asignar un elemento o símbolo para indicar que se trata de un valor desconocido. Es, por tanto, una misión del proceso de $\mathcal{KDD}$ establecer y aplicar unas técnicas que corrijan estas ausencias antes de aplicar el algoritmo de minería.

La técnica más simple de tratamiento de valores ausentes consiste en eliminar las instancias donde alguno de los atributos que la componen incluya un valor ausente. Esta técnica, aunque muy simple, en la mayoría de las ocasiones no es la más aconsejable ya que conlleva una enorme perdida de información del resto de atributos de la instancia que sí disponen de valor.

Una estrategia que soluciona la pérdida de instancias se basa en el uso de métodos estadísticos o heurísticos para determinar un posible valor para los atributos ausentes. En este sentido, las técnicas más conocidas son aquellas que utilizan la media para dominios cuantitativos, y la moda para dominios discretos, como estimadores para determinar los valores ausentes. Estas técnicas tienen un inconveniente si el número de valores ausentes de un atributo es muy elevado debido a que generarían el mismo valor para todas las ausencias. Para resolver ese proble-

ma, es necesario que los valores de la media y la moda sean generados a partir de diferentes subconjuntos de las instancias.

4.4.2. Traslación del dominio de los atributos

La mayoría de los algoritmos de aprendizaje distinguen entre atributos continuos y discretos. En la resolución de ciertos problemas, los algoritmos de minería de datos requieren que el dominio de los atributos sea de un determinado tipo que, a veces, no coincide con en el que se encuentra almacenado en la base de datos. Así, muchos algoritmos de aprendizaje trabajan únicamente con atributos discretos, por lo que resulta necesario realizar una traslación del dominio original del atributo al dominio permitido por el algoritmo. Básicamente existen dos tipos de traslaciones: de atributos discretos a continuos, denominada *numeración*, y de atributos continuos a discretos, denominada *discretización*.

Numeración

La numeración consiste en la trasformación de un dominio no ordenado (discreto) en un dominio ordenado (numérico). No suele ser un proceso frecuente, aunque existen muchos trabajos referentes a este campo. Algunos de los más significativos pueden consultarse en [Breiman *et al.*, 1984] [Hampson and Volper, 1986] [Utgoff and Brodley, 1990] [Van de Merckt, 1992] [Van de Merckt, 1993].

El método más simple para realizar la numeración de atributos discretos consiste en asignar un valor numérico creciente, en base a algún conocimiento sobre la realidad del atributo, a cada uno de los posibles valores del atributo. Este método, aunque es un método sencillo, presenta el inconveniente de que esta ordenación artificial produzca posteriormente errores en la interpretación de los resultados.

Discretización

El proceso de discretización consiste en la traslación de un dominio numérico a un dominio discreto. Dicho de otra forma, se trata de encontrar puntos de corte que dividan el espacio de valores continuos en una serie de intervalos, pudiéndolos tratar, a partir de ese momento, como atributos discretos con un número finito

de valores. Este proceso es bastante frecuente puesto que una gran mayoría de algoritmos de aprendizaje trabajan únicamente con dominios discretos.

En general, los métodos de discretización suelen clasificarse en *métodos de discretización supervisada* y *métodos de discretización no supervisada*. Esta distinción se realiza dependiendo de si se utiliza o no información sobre las clases para agrupar los datos. Los métodos supervisados emplean información sobre la clase a la que pertenece cada caso del conjunto de entrenamiento a la hora de evaluar y escoger los puntos de corte que definen los intervalos en que quedan agrupados los valores numéricos. Los métodos no supervisados no tienen en cuenta esta información y utilizan la distribución de valores de un atributo continuo como única fuente de información.

La forma más simple de discretización consiste en crear un número preestablecido de intervalos. Las técnicas más habituales dentro de la discretización no supervisada consisten en dividir el rango del atributo en k intervalos del mismo tamaño o amplitud (*Equiwidth*, discretización de igual anchura), o bien asignar a cada intervalo el mismo número de valores (*Equidepth*, discretización de igual frecuencia). En [Dougherty *et al.*, 1995] se repasan distintos métodos de discretización utilizados en inteligencia artificial y aprendizaje automático.

Por su parte, los métodos de discretización supervisada utilizan información discriminatoria de las instancias para determinar los puntos frontera. El discretizador 1R de Holte [Holte, 1993] es una variante supervisada de este tipo de métodos, que ajusta los límites de los intervalos en función de la clase a la que pertenezcan los casos del conjunto de entrenamiento.

En [Liu and Setiono, 1995] se presenta un método estadístico, denominado Chi^2, que utiliza las frecuencias relativas de las clases entre intervalos adyacentes, de tal forma que si son muy similares es posible su fusión. La fórmula para calcular el valor de χ^2 es $\sum_{i=1}^{2}\sum_{j=1}^{|C|}\frac{(A_{ij}-E_{ij})^2}{E_{ij}}$, donde $A_{ij}=\frac{R_iC_j}{n}$, siendo $|C|$ el número de clases, A_{ij} el número de instancias en el intervalo i con clase j, E_{ij} la frecuencia esperada de A_{ij}, R_i el número de instancias en el intervalo i, C_j el número de instancias de la clase j en los dos intervalos y n en número de instancias en los dos intervalos.

La teoría de la información también ha originado la aparición de algunas técnicas de discretización, como el uso de la ganancia de información o del criterio de

proporción de ganancia en la construcción de árboles de decisión. El método de discretización supervisada de Fayyad e Irani [Fayyad and Irani, 1993], basado en el principio MDL de Rissanen [Rissanen, 1983], es un ejemplo destacado de este tipo de métodos. El algoritmo decide dónde colocar los cortes de los intervalos dependiendo de la entropía de la clase. El método se aplica recursivamente a los dos intervalos obtenidos tras cada división hasta satisfacer una condición de parada. En este caso, la condición de parada es el citado Principio de Longitud de Descripción Mínima. Una alternativa similar se presenta en [Catlett, 1991], pero en este caso el algoritmo termina cuando el número de ejemplos en un intervalo es suficientemente pequeño o cuando se han creado un número máximo de intervalos.

4.4.3. Selección de atributos relevantes

La selección de atributos relevantes (*feature selection*) es un procedimiento fundamental durante el proceso de $\mathcal{KDD}$ para que el algoritmo de minería pueda realizar su función con mayor probabilidad de éxito. El objetivo general es reducir el conjunto de datos original, eliminando atributos con información redundante o irrelevante con el objetivo de facilitar el proceso de aprendizaje posterior. Por tanto, consiste en identificar y suprimir del conjunto de datos aquellos atributos con escasa o nula relevancia dentro del mismo.

Existen diferentes enfoques para determinar la relevancia de atributos. Entre ellos se describen someramente el algoritmo RELIEFF y una técnica basada en el criterio de información mutua.

RELIEFF

El algoritmo RELIEFF, descrito en [Kononenko, 1994] [Kononenko and Simec, 1995], es una ampliación del algoritmo RELIEF, desarrollado por Kira y Rendell [Kira and Rendell, 1992]. La mejora del trabajo de Kononenko consiste en permitir valores ausentes y más de dos clases en el conjunto original.

La relevancia de los atributos se determina analizando las instancias más cercanas. Así, dada una instancia, RELIEF busca a sus dos instancias más cercanas, una de la misma clase y otra de una clase diferente, y actualiza los pesos de los atributos involucrados dependiendo de si sus valores son iguales o no al ejemplo.

La idea del algoritmo es favorecer los atributos que tengan valores diferentes en instancias parecidas de diferente clase y valores iguales en instancias parecidas de la misma clase.

Técnicas basadas en el criterio de información mutua

La técnica de selección de atributos relevantes basada en el criterio de información mutua [Cover and Thomas, 1991] tiene como objetivo determinar la relación entre los atributos y la clase, entendiendo el concepto de relación como la información que los primeros ofrece sobre el segundo. Así, aquellos atributos que no ofrezcan información de la clase podrían ser eliminados.

$$I(\mathcal{A}_i, \mathcal{C}) = \sum_{k}^{K} \sum_{j}^{J} P(c^j \wedge \mathcal{A}_i^k) \frac{P(c^j \wedge \mathcal{A}_i^k)}{P(c^j) P(\mathcal{A}_i^k)} \tag{4.1}$$

Para ello, la información I, entre un atributo $\mathcal{A}_i$ y la clase C, puede calcularse mediante la ecuación 4.1, donde $P(c^j)$ es la probabilidad de que la clase C tome el valor j, $P(\mathcal{A}_i^k)$ la probabilidad de que el atributo $\mathcal{A}_i$ tome el valor k, $P(c^j \wedge \mathcal{A}_i^k)$ la probabilidad de que ocurran los dos sucesos conjuntamente, K el número de valores distintos del atributo A_i y J el número de clases.

Además, para determinar la información mutua entre atributos se genera una matriz donde la celda (i, j) almacena $I(\mathcal{A}_i, \mathcal{A}_j)$. Si para cualquier fila i existe un elemento tal que $t_{i,i} = t_{i,j}$, entonces el atributo $\mathcal{A}_j$ es redundante con respecto al $\mathcal{A}_i$ y se puede eliminar.

4.4.4. Reducción de instancias

Otro de los procedimientos fundamentales durante el proceso de $\mathcal{KDD}$ es la reducción de instancias del conjunto inicial. El problema de la reducción de instancias surge cuando las técnicas de minería de datos se ven incapaces de abordar el proceso de aprendizaje sobre un conjunto de datos con un elevado número de instancias.

En general, el proceso de reducción consiste en encontrar un subconjunto S del conjunto inicial $\mathcal{D}$ que contenga el mismo conocimiento. Esto es, la técnica de reducción debe preservar el conocimiento del conjunto original de datos, de forma

que el algoritmo de minería pueda obtener lo mismos patrones desde el nuevo conjunto de datos.

En función de cómo se realice el proceso de búsqueda del subconjunto S, las técnicas de reducción de instancias pueden clasificarse en: incrementales, decrementales y por lotes.

La búsqueda incremental comienza con un subconjunto S vacío al que se le van añadiendo instancias del conjunto original de datos que cumplan un determinado criterio preestablecido. Algunos de los algoritmos incrementales más significativos son el algoritmo de Hart, denominado CNN (*Condensed Nearest Neighbor*) [Hart, 1967] y los algoritmos IBL2, IBL3, IBL4 e IBL5 (*Instance-Based Learning*) [Aha *et al.*, 1991].

La búsqueda decremental comienza con el conjunto original de instancias ($S = \mathcal{D}$), y se van eliminando de acuerdo a algún criterio establecido. Dentro del grupo de algoritmos decrementales están: RNN (*Reduced Nearest Neighbor*) [Gates, 1972], ENN (*Edited Nearest Neighbor*) [Wilson, 1972] y SNN (*Selective Nearest Neighbor*) [Ritter *et al.*, 1975] [Lindenbaum *et al.*, 1999].

Por último, la búsqueda por lotes establece un criterio de reducción y analiza el conjunto de datos completo eliminando todas aquellas instancias que satisfacen dicho criterio. Las técnicas más comunes que utilizan esta estrategia de búsqueda son AENN (*All-kNN Edited Nearest Neighbor*) y UENN (*Unlimited Edited Nearest Neighbor*) [Tomek, 1976].

4.5. Minería de datos

Una vez que los datos han sido preparados y, siguiendo la secuencia descrita para el proceso de $\mathcal{KDD}$, la siguiente etapa del proceso es la minería de datos. En esta fase y en las sucesivas hay que tomar algunas decisiones importantes:

1. Elegir la técnica de minería de datos que se va aplicar. Esta elección depende, en gran medida, del tipo de conocimiento o patrón que se desea descubrir.

2. Elegir el algoritmo más apropiado para implementar dicha técnica.

3. Establecer los criterios de evaluación de los modelos obtenidos para mostrar únicamente la información útil y válida.

4. Elegir el sistema de interpretación y visualización de los resultados obtenidos atendiendo, principalmente, a su sencillez de entendimiento.

Tanto la técnica elegida como el algoritmo necesario para desarrollarla dependerán de la tarea que necesitemos llevar a cabo con los datos ya preparados. En [Hernández Orallo *et al.*, 2004] se hace una diferenciación clara entre tareas y métodos. Se entiende como tarea de la minería de datos el tipo de problema que se pretende resolver. El método, por su parte, será la forma en la que llevamos a cabo la tarea.

Las tareas se dividen en dos grandes clases:

Predictivas Se parte de un conjunto de ejemplos que llevan asociados una clase. La tarea consiste en predecir uno o más valores desconocidos o futuros para uno o más ejemplos. Las dos tareas predictivas más habituales son:

- Clasificación. El objetivo de la clasificación es aprender una función $\lambda : E \rightarrow S$, donde E es el conjunto de valores de entrada para los ejemplos disponibles, y S es el conjunto de valores de salida. El conjunto S es nominal, es decir, puede tomar un conjunto de valores finitos, a los que se denomina clases. La función λ representa la correspondencia entre los ejemplos, es decir, hay un único valor de S para cada valor de E. Dicha función será capaz de determinar la clase de un nuevo ejemplo sin etiquetar. Debido a que será dicho tarea la desarrollada en la presente tesis, abordaremos sus principales características en el tema 5.
- Regresión. Las técnicas de regresión permiten hacer predicciones sobre los valores de cierta variable Y (dependiente), a partir de los de otra X (independiente), entre las que intuimos que existe una relación. Mediante las técnicas de regresión de una variable Y sobre una variable X, buscamos una función que sea una buena aproximación de una nube de puntos (x_i, y_i), mediante una curva del tipo $\hat{Y} = f(X)$. Para ello hemos de asegurarnos de que la diferencia entre los valores y_i e $\hat{y}_i$ sea tan pequeña como sea posible.

  Mediante las técnicas de regresión inventamos una variable $\hat{Y}$ como función de otra variable X. Este hecho se denomina *relación funcional.*

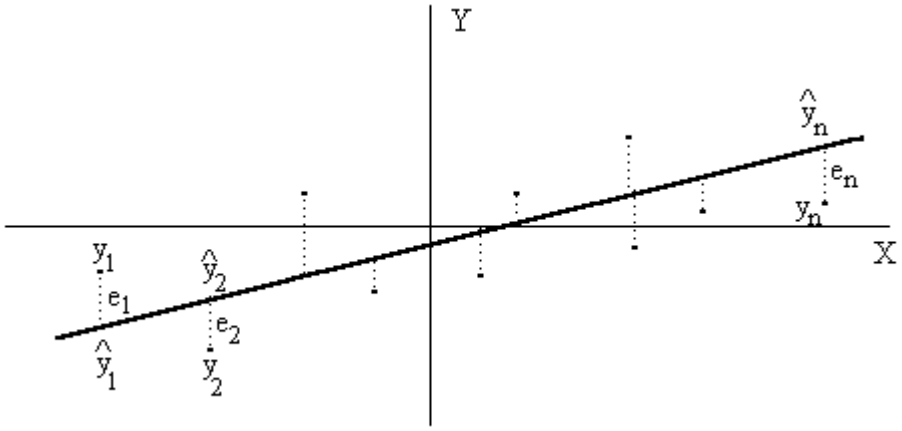

Figura 4.3: Regresión de Y sobre X

El criterio para construir $\hat{Y}$, tal como citamos anteriormente, es que la diferencia entre Y e $\hat{Y}$ sea pequeña, es decir: $Y - \hat{Y} =$ error.

El término que hemos denominado *error* debe ser tan pequeño como sea posible. El objetivo será buscar la función o modelo de regresión $\hat{Y} = f(X)$ que lo minimice. En la figura 4.3 se muestra una recta de regresión de la variable Y sobre la variable X donde los errores a minimizar son las cantidades $e_i^2 = (y_i - \hat{y}_i)^2$.

Descriptivas El conjunto de ejemplos disponibles se encuentra sin ningún tipo de orden ni etiqueta que los defina. El objetivo es detectar regularidades en los datos de cualquier tipo: agrupaciones, contornos,asociaciones, valores anómalos, etc. Las tareas descriptivas más comunes son:

- Agrupamiento (clustering). El clustering o agrupamiento es una técnica similar a la clasificación en el sentido de organizar los datos en clases o categorías. La principal diferencia es que la clasificación no está dictada por las clases de las instancias, puesto que éstas son desconocidas en el conjunto de entrenamiento. El objetivo, por tanto, es descubrir la distribución y relación oculta entre las instancias de un conjunto de datos [Jain and Dubes, 1988] [Arabie *et al.*, 1996].

  Definición 4.8. **Clustering**. Dado un conjunto de datos $\mathcal{D}$, denominado conjunto de entrenamiento, formado por un conjunto $\mathcal{I}$ de ins-

tancias, $\mathcal{I} = i_1, i_2, ..., i_n$, caracterizadas por un conjunto $\mathcal{A}$ de atributos, $\mathcal{A} = a_1, a_2, ..., a_m$, el objetivo del clustering consiste en particionar el conjunto $\mathcal{I}$ en k-conjuntos o *clusters* de forma que optimicen un determinado criterio de similitud entre los atributos. Este criterio de optimización puede consistir en maximizar la distancia entre clusters (*intercluster*) o minimizar la distancia entre las instancias de un mismo cluster (*intracluster*).

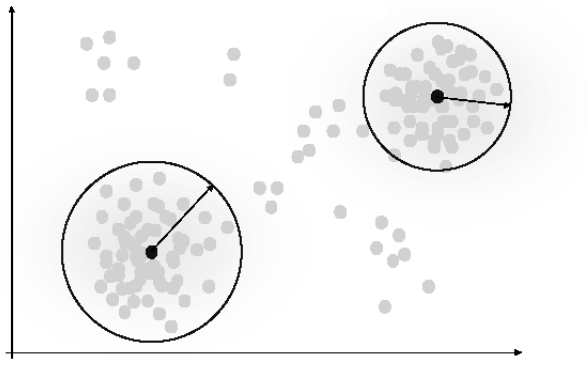

Figura 4.4: Ejemplo de Clustering

- Correlaciones y factorizaciones. Esta tarea se centra exclusivamente en atributos numéricos. El objetivo es ver si dos o más atributos numéricos del conjunto de ejemplos están relacionados linealmente o de algún otro modo. Este tipo de relaciones son bidireccionales, es decir, no orientadas.

- Reglas de asociación. El objetivo del proceso de extracción de *reglas de asociación* consiste, básicamente, en descubrir si la presencia de un conjunto de items, que se encuentran en un número determinado de transacciones, puede ser utilizada para inducir la presencia de otro conjunto de items en las mismas transacciones.

  Una *regla de asociación* se expresa en forma de implicación $C_1 \Rightarrow C_2$, donde C_1 y C_2 son conjuntos de atributos o items. Es decir, las *reglas de asociación* indican la probabilidad de que se encuentre, entre las transacciones de una base de datos, un determinado conjunto de items cuando también se encuentra otro conjunto distinto de items.

> ***Definición 4.9.*** Sea $\mathcal{A} = \{A_1, A_2, ..., A_n\}$ el conjunto de todos los atributos de una base de datos $\mathcal{D}$. Una **regla de asociación** es una implicación $C_1 \Rightarrow C_2$, donde $C_1 \subseteq \mathcal{A}$, $C_2 \subseteq \mathcal{A}$ y $C_1 \cap C_2 = \oslash$. C_1 se denomina *antecedente* de la regla y C_2 *consecuente*.

Para resolver dichas tareas se dispone de una gran variedad de métodos o técnicas. Ninguno de dichos métodos esta unívocamente relacionado con una tarea y viceversa. Por el contrario, para cada una de las tareas expuestas tenemos una gran variedad de métodos que pueden ser aplicados, y cada uno de dichos métodos puede resolver, en ciertos aspectos, muchas de las tareas expuestas.

A continuación expondremos brevemente la relación de técnicas más usadas. El estudio en detalle de dichas técnicas se escapa del ámbito de la presente memoria de tesis, por tanto tan sólo haremos una somera descripción de cada una de ellas.

Técnicas estadísticas

Puede considerarse a la estadística como la técnica pionera para obtener información interesante de los datos. Buscan formar modelos a partir de fórmulas algebraicas y funciones lineales o no lineales. Suelen recurrir al uso de estadísticos descriptivos como la media, varianza, correlaciones, etc. Dependiendo en la forma de conseguir el patrón se pueden distinguir entre técnicas paramétricas y no paramétricas.

Las técnicas paramétricas obtienen un patrón estimando unos parámetros o coeficientes de un modelo ya predeterminado. En este grupo de técnicas se engloban la regresión lineal, la regresión logarítmica y la regresión logística. Los discriminantes lineales y no lineales forman parte también de este grupo de técnicas.

Las técnicas no paramétricas son más flexibles, permitiendo el modelado de fenómenos más complejos. La regresión no paramétrica puede realizarse estimando la función de regresión mediante el ajuste local de modelos paramétricos, métodos basados en series ortogonales, suavizado mediante splines o alguna técnica de aprendizaje supervisado como las redes neuronales, los k-vecinos más cercanos o los árboles de regresión. Por su parte, la discriminación no paramétrica usa habitualmente el estimador núcleo, que tiene la propiedad de distribuir el peso $1/n$ de cada dato observado en un entorno suyo de forma continua.

Ambos tipos de técnicas son más apropiadas para problemas numéricos, aunque aplicando numeración pueden ser usados también con datos categóricos. Las técnicas no paramétricas suelen ser menos comprensibles debido al uso habitual de funciones núcleo, aunque por contra, son más flexibles y precisas. Para grandes volúmenes de datos las técnicas paramétricas suelen ser mucho más eficientes.

Técnicas bayesianas

La base fundamental de estas técnicas es el teorema de Bayes, mediante el cual las observaciones o nuevas evidencias, son usadas para actualizar o inferir la probabilidad de que la hipótesis sea cierta. Los modelos de representación de conocimiento con incertidumbre se consiguen mediante las redes de bayes.

Una Red Bayesiana consta de dos componentes. El primero de ellos, más cualitativo, está representado por un grafo acíclico dirigido $G = (V, E)$ donde los nodos (el conjunto finito V) son variables aleatorias del problema, y los arcos ($E \subset V \times V$) indican relaciones entre variables. El segundo de ellos, cuantitativo, se trata de un conjunto de distribuciones de probabilidad condicionadas (una por nodo) donde la distribución en cada nodo está condicionada al posible valor de cada uno de los padres. En definitiva, es un modelo probabilístico multivariado que relaciona un conjunto de variables aleatorias mediante un grafo dirigido, el cual indica explícitamente influencia causal.

El clasificador más relevante que se basa en estas técnicas es el clasificador de Naïve Bayes [Duda and Hart, 1973]. El clasificador de Naïve Bayes es el caso más simple de clasificación con redes bayesianas. En este clasificador, la estructura de la red es fija, y tan sólo se necesitan aprender los parámetros (probabilidades). Su fundamento principal es la suposición de que todos los atributos son independientes, conocido el valor de la variable clase. Esto da lugar a un modelo gráfico probabilístico en el que el único nodo raíz es la clase, y los atributos son todos nodos hijos del raíz.

Tomando como base el clasificador de Naïve Bayes, existen clasificadores que pretenden mejorar sus resultados, como por ejemplo TAN (Tree Augmented Naïve Bayes) [Friedman *et al.*, 1997] , que para mejorar la tasa de aciertos permite cierta dependencia entre los atributos o BAN (Bayesian Network Augmented Naïve Bayes), que aprende una red bayesiana para los atributos sin la clase y posterior-

mente se aumenta el modelo añadiendo la variable clase y aristas desde la clase a los atributos.

Las técnicas bayesianas son fáciles de usar, eficientes y permiten trabajar con gran cantidad de atributos. Suelen ser muy robustos al ruido, pero su gran problema es el gran coste computacional necesario para llevarlas a cabo.

Árboles de decisión

Un árbol de decisión es una de las representaciones más sencilla de representar el conocimiento. Principalmente se utiliza para construir clasificadores fáciles de entender por el ser humano. Se basa en la lógica de proposiciones extendida y utiliza la técnica *divide y vencerás* para clasificar el conjunto de ejemplos en un número finito de clases predefinidas realizando particiones o subconjuntos de forma recursiva.

Existe una serie de algoritmos desarrollados para la construcción de árboles de decisión, entre ellos CLS (*Concept Learning System*) [Hunt *et al.*, 1966], ID3 (*Induction Decision trees*) [Quinlan, 1979], CART (*Classification and Regression Trees*) [Breiman *et al.*, 1984], OC1 (*Oblique Classifier 1*) [Murthy *et al.*, 1994], ASSISTANT [Cestnik *et al.*, 1987], C4.5 [Quinlan, 1993b]. Muchos de estos desarrollos se han convertido en herramientas comerciales, como RuleMaster (1984), Ex-Tran (1984), Expert-Ease(1983) o C5/See5 (2000).

La técnica de árbol de decisión toma como entrada un conjunto de datos $\mathcal{D}$, compuesto por una serie de atributos $X = x_1, x_2, ..., x_m$ y una clase $\mathcal{C}$ con valores $c_1, c_2, ..., c_l$ predefinidos e induce una estructura en forma de árbol de la siguiente manera:

1. Se crea un nodo raíz $\mathcal{N}$ con todos los ejemplos.
2. Si todos los elementos de $\mathcal{N}$ son de la misma clase, se ha encontrado una solución y el subárbol se cierra.
3. Se elige una condición de partición siguiendo un criterio de partición (*split criterion*).
4. El problema queda subdividido en dos subárboles, los que cumplen la condición y los que no la cumplen. Se vuelve a aplicar el paso 2 para cada uno de los dos subárboles.

Al final del proceso, los nodos del árbol quedan etiquetados con nombres de atributos, las ramas con los valores posibles para el atributo, y las hojas con las diferentes clases. Una vez se obtiene el árbol a partir del conjunto de entrenamiento, los ejemplos del test cuya clase se desconoce son clasificados recorriendo el árbol por aquellas ramas para las cuales los valores de sus atributos se cumplen hasta alcanzar una hoja.

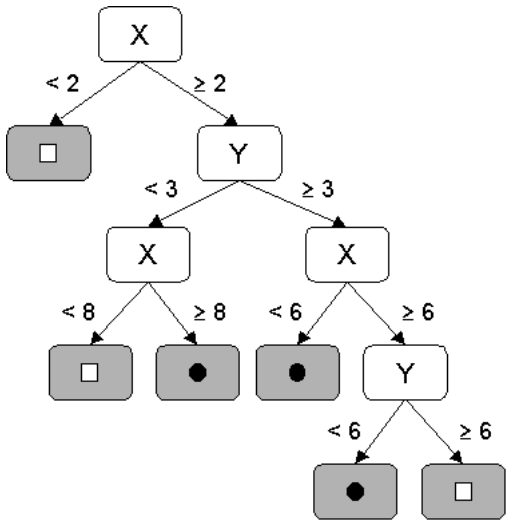

Figura 4.5: Ejemplo de árbol de decisión

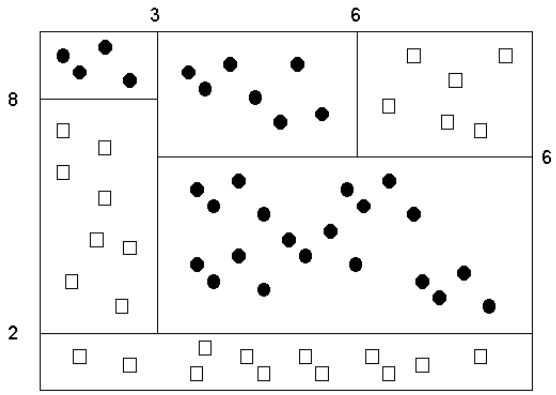

Figura 4.6: Cortes realizados según el criterio de partición

La figura 4.5 se muestra un ejemplo de árbol de decisión para un conjunto de entrenamiento formado por dos atributos numéricos y dos clases, mientras que en la figura 4.6 se representa el conjunto de entrenamiento con los cortes utilizados para clasificar los ejemplos.

Los sistemas basados en árboles de decisión forman una familia llamada TDIDT (*Top-Down Induction of Decision Trees*), cuyo representante más conocido es C4.5 [Quinlan, 1993b], que permite clasificar ejemplos con atributos que toman valores continuos.

La principal ventaja de esta representación son sus reducidos requisitos computacionales. Sin embargo cuando el número de atributos es elevado y para estos lo es también el dominio de valores, se vuelven difíciles de comprender. La construcción y clasificación de un árbol de decisión viene dada por: la función de selección, el criterio de parada y el algoritmo de podado.

La función de selección es el criterio utilizado por el árbol para particionar el conjunto de entrenamiento, esto es, obtener los sucesivos subárboles que lo componen. Las mas utilizada es la ganancia de información media, que selecciona para cada partición el atributo que genera la reducción máxima de la entropía media. Otras medidas utilizadas son GINI y la ganancia de información normalizada.

El criterio de parada determina cuando una partición se completa. Generalmente se llega a una hoja cuando todos los ejemplos del subconjunto que contiene pertenecen a la misma clase, pero no es el único criterio.

El algoritmo de podado intenta prevenir el sobreajuste, que puede evitarse mediante un pre-podado o un post-podado donde el árbol ya esta totalmente construido. Algunas técnicas son el *Error-Complexity*, el error pesimista o el error mínimo.

Las principales diferencias entre los distintos algoritmos de árboles de decisión radica en la elección de la función de selección y en la estrategia de podado.

Los árboles de decisión son fáciles de usar, admiten atributos numéricos y categóricos, y tratan bien los atributos no significativos, el ruido y los valores ausentes. Su principal ventaja es la comprensibilidad, y su peor cualidad la falta de precisión y que son inestables ante variaciones de la muestra.

Técnicas relacionales, declarativas y estructurales

Estas técnicas presentan los modelos mediante lenguajes declarativos, como los lógicos y funcionales. Estos métodos permiten descubrir patrones complejos,

haciendo uso de la estructura de los propios datos y las relaciones entre ellos. La técnica más representativa es la programación lógica inductiva (ILP[1]).

En 1990, Sthephen Muggleton define la programación lógica inductiva como la intersección de la programación lógica y el aprendizaje inductivo [Muggleton, 1991]. En su formulación más simple, la tarea de la ILP consiste en aprender la definición de un predicado a partir de hechos que son ciertos para este predicado, hechos que son falsos para este predicado y la definición de otros predicados auxiliares que se pueden usar en el proceso de aprendizaje. Es decir, a partir de una base de datos de hechos y resultados esperados, que se dividen en verdaderos y falsos, un sistema ILP intenta crear un programa lógico que demuestre todos los resultados ciertos y ninguno de los falsos.

Aunque la ILP es la aproximación más extendida, no es la única existente, y hay diversos métodos alternativos, como el aprendizaje basado en grafos [Washio and Motoda, 2003], los modelos relacionales probabilísticos [Koller and Pfeffer, 1998], los árboles de decisión relacionales (como SCART [Kramer and Widmer, 2000] o las reglas de asociación relacionales [Dehaspe and Toivonen, 1999].

Las técnicas que usan estos lenguajes forman un conjunto denominado *Minería de Datos Relacional*. Son técnicas muy precisas, que permiten capturar patrones relacionales y recursivos y los modelos generados son comprensibles. Por contra, son difíciles de manejar y los métodos existentes no son muy eficientes.

Redes neuronales artificiales

Los sistemas de aprendizaje basados en redes neuronales utilizan modelos matemáticos fundamentados en los conceptos del funcionamiento del sistema nervioso de los seres vivos [McCulloch and Pitts, 1943], mediante una simulación simple del proceso de aprendizaje de las neuronas del cerebro.

Para ello, cada neurona es simulada mediante una unidad de cálculo que se conecta, mediante capas, con otras neuronas de tal forma que la salida de una es la entrada de otra. La primera capa se conoce como capa o vector de entrada ya que será la que tome los datos externos (el conjunto de datos) y la última capa se denomina capa o vector de salida, ya que ofrecerá los resultados del procesamiento.

[1]del inglés Inductive Logic Programming

El resto de capas se denominan capas o vectores ocultos. En la figura 4.7 se ofrece una representación de una red neuronal.

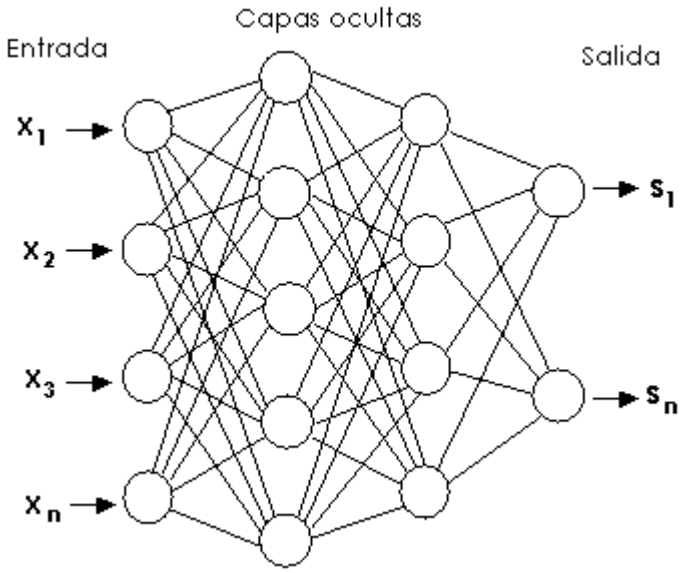

Figura 4.7: Estructura de una red neuronal

En la práctica, para el diseño de la red neuronal es necesario decidir una serie de parámetros, como el número de unidades (neuronas), la interconexión entre ellas, el algoritmo de aprendizaje y la codificación de las entradas y salidas.

Los distintos sistemas de redes neuronales existentes abordan diferentes tareas. Los más extendidos dentro de cada tipo son: las memorias asociativas, las redes de Hopfield, la máquina de Bolzman y la máquina de Cauchy, las redes de aprendizaje competitivo, las redes de Kohonen o mapas auto-organizados [Kohonen, 1982] [Kohonen, 1984] y las redes de resonancia adaptativa (ART) para tareas descriptivas, la red Adaline, el perceptrón multicapa [Rosenblatt, 1958] y la memoria asociativa bidireccional para tareas predictivas y las redes de función de base radial (RBF, *Radial Basic Function*) [Moody and Darken, 1989], que son modelos híbridos que tratan de aprovechar las ventajas de ambas estrategias.

Obtienen en general precisiones muy altas cuando están bien ajustadas. Son muy precisas y capturan modelos no lineales, aunque son muy sensibles a valores anómalos, necesitan muchos ejemplos y suelen ser lentas además de generar modelos no comprensibles. Se necesita además cierta experiencia para ajustar el número de capas y nodos internos.

Máquinas de soporte vectorial

Estas técnicas intentan maximizar el margen entre los grupos o clases formadas. Se basan para conseguirlo en transformaciones que aumentan la dimensionalidad. Las funciones usadas con tal fin son denominadas funciones *núcleo*. Según la función núcleo usada y la forma de trabajar con el margen existen muy diversas opciones.

Las maquinas de soporte vectorial pertenecen al grupo de los clasificadores lineales, puesto que inducen separadores lineales o hiperplanos en espacios de características de muy alta dimensionalidad. La idea que hay detrás de las SVM[2] es encontrar el hiperplano separador que está equidistante de los ejemplos más próximos de cada clase. De forma intuitiva, dicho hiperplano está situado en la posición más neutra posible con respecto a las clases representadas por el conjunto de datos, sin estar sesgado hacia ninguna de las clases.

A nivel algorítmico, el aprendizaje de las SVM representa un problema de optimización con restricciones que puede ser resuelto usando técnicas de programación cuadrática (QP).

Cuando las clases no son linealmente separables, el aprendizaje de separadores no lineales se consigue mediante una transformación no lineal del *espacio de atributos de entrada* en un *espacio de características* de dimensionalidad mucho mayor y donde sí existe separación lineal de las clases. Para trabajar de manera eficiente en el espacio de características, sin necesidad de calcular explícitamente las transformaciones de los ejemplos de aprendizaje se recurre a las **funciones núcleo**, que permiten calcular el producto escalar de dos vectores en el espacio de características. Por tanto, un requisito básico para el éxito en el aprendizaje es la elección de una función núcleo adecuada, que refleje a priori el conocimiento del problema a tratar.

Las maquinas de soporte vectorial permiten trabajar con datos de dimensiones grandes y proporcionan modelos muy precisos. El principal problema es la elección de la función núcleo, y que al transformar los atributos iniciales , los modelos generados no son comprensibles.

[2]del inglés Support Vector Machine

Técnicas evolutivas y difusas

Este tipo de técnicas forman, junto con las redes neuronales, el conjunto de técnicas conocidas como soft computing (que traducida sería algo así como computación flexible). Están formados por los métodos genéticos, junto con las pertenencias difusas (fuzzy).

La principal característica de los métodos de computación flexible es su tolerancia a la imprecisión y a la incertidumbre, lo que permite que se adapten muy bien a entornos cambiantes. Puesto que las redes neuronales han sido tratadas de forma independiente en un apartado previo, nos ocuparemos de los otros dos pilares de la computación flexible: los algoritmos evolutivos y la lógica difusa.

La computación evolutiva la forman un conjunto de algoritmos estocásticos de búsqueda basados en abstracciones del proceso de evolución de *Darwin*. En este area se han presentado distintos modelos computacionales denominados algoritmos evolutivos: algoritmos genéticos, estrategias de evolución, programación evolutiva y programación genética. Dichos algoritmos tratan de modelar la evolución siguiendo tres características comunes:

1. Se usa el proceso de aprendizaje colectivo de una población de individuos

2. Los descendientes de los individuos se generan mediante procesos no determinísticos que tratan de modelar los procesos de mutación y cruce. La mutación representa la auto-replicación errónea de los los individuos, mientras el cruce representa el intercambio de material genético de dos o más individuos.

3. Se asigna una medida de calidad, llamada *medida de adaptación o fitness* a cada individuo mediante el proceso de evaluación. El *operador de selección* favorece en el proceso de evolución a los mejores individuos.

Una de las características importantes de los algoritmos evolutivos es que realizan una búsqueda global. Al trabajar con una población de soluciones candidatas en vez de con una individual y los operadores estocásticos reducen el riesgo que caer en un óptimo local.

Por su parte, la lógica difusa [Zadeh, 1965] permite modelar conocimiento impreciso y cuantitativo así como transmitir, manejar incertidumbre y soportar el razonamiento humano de una forma natural.

Los sistemas basados en reglas que usan conjuntos difusos para describir el valor de sus variables se denominan sistemas basados en reglas difusas, donde los términos lingüísticos de las reglas son caracterizados mediante conjuntos difusos. En los conjuntos difusos, la función de pertenencia asigna a cada elemento un grado de pertenencia dentro del intervalo [0,1]. Esto permite representar conceptos con límites borrosos, mal definidos, pero con significado definido de forma completa y precisa.

El interés de la lógica difusa en la minería de datos se deriva de la utilidad de recoger la imprecisión y flexibilidad de los límites a la hora de describir e identificar los patrones. Tienen numerosos usos, pero se han usado frecuentemente en Agrupamiento [Turksen, 1998], clasificación [Janikow, 1998] y reglas de asociación [Chen and Wei, 2002].

Las técnicas evolutivas son bastante flexibles, al poder cambiar el heurístico sin modificar el algoritmo. Por contra, su coste computacional en tiempo y almacenamiento suele ser muy altos. Con las técnicas difusas se puede conseguir gran precisión, debido a la flexibilidad de las fronteras establecidas.

Técnicas basadas en casos, vecindad y distancia

Son métodos que se basan en medir la distancia al resto de elementos. Esto puede hacerse de forma directa o mediante el uso de funciones más complejas como las de densidad.

La técnica de clasificación más popular basada en distancia es la de los vecinos más cercanos. En esta técnica, las instancias del conjunto de datos se almacenan en función de su similitud. Para clasificar una nueva instancia, se extraen aquellas más parecidas y se clasifica la nueva en base a ellas. Así, el proceso de aprendizaje es trivial mientras que el proceso de clasificación es el que consume el mayor tiempo.

Las técnicas para agrupamiento por distancias crean grupos o *clusters* de elementos "próximos" entre sí. Dependiendo de la estrategia utilizada a la hora de realizar el agrupamiento, estos algoritmos pueden ser clasificados en:

- **Métodos basados en particiones.** Construyen k particiones en los datos y cada partición representa un grupo o cluster. Estos métodos comienzan creando una única partición inicial e iteran hasta cumplir un criterio de

parada. Algunos de los algoritmos que siguen esta estrategia son: k-means, PAM (*Partitioning Around Medoids*), CLARA (*Clustering Large Applications*) y CLARANS (*Clustering large Applications based upon RANdomized Search*).

- **Métodos jerárquicos.** Los datos se agrupan de manera arborescente. Se dividen en *aglomerativos* y *por división*. El primero, también denominado *bottom-up*, empieza con un cluster por cada instancia del conjunto de datos y va uniendo los clusters más parecidos hasta obtener k clusters. El algoritmo AGNES (*Agglomerative Nesting*) [Kaufman L., 1990] es un ejemplo de esta forma de agrupamiento.

  El segundo método, también denominado *top-down*, empieza con un único cluster para todas los instancias y lo va dividiendo en clusters más pequeños hasta obtener los k clusters deseados. Un ejemplo de método divisorio es el algoritmo DIANA (*Divisive Analysis*) [Kaufman L., 1990]).

- **Métodos basados en densidades.** Establecen un nivel de vecindad y agrupan las instancias que pertenecen a ese nivel. DBSCAN (*Density-Based Spatial Clustering of Applications with Noise*) y DENCLUE (*DENsity-based CLUstEring*) [Hinneburg and Keim, 1998] utilizan este método.

- **Métodos basados en rejillas.** Dividen el espacio de búsqueda en rejillas a diferentes niveles y utilizan esta división para agrupar las instancias. Entre los algoritmos que utilizan este método están STING (*STatistical INformation Grid*) [Wang *et al.*, 1997] y CLIQUE (*Clustering In QUEst*) [Agrawal *et al.*, 1998].

- **Métodos basados en modelos.** Buscan modelos que se ajusten con las instancias de cada cluster, como por ejemplo EM (*Expectation Maximization*) [Dempster *et al.*, 1977] y AutoClass (*Automatic classification or clustering*) [Cheeseman and Stutz, 1996].

Un caso algo distinto, pero siguiendo la filosofía de la similitud de instancias lo constituye el razonamiento basado en casos [Sycara *et al.*, 1991]. Parte de la misma filosofía que los vecinos más cercanos, pero los ejemplos no son vectores de números reales, sino descripciones simbólicas más complejas que, por tanto, deben

Nombre	Predictivo		Descriptivo		
	a	b	c	d	e
Técnicas Estadísticas	✓	✓		✓	✓
Técnicas Bayesianas	✓				
Árboles de Decisión	✓	✓	✓	✓	
Técnicas relacionales, declarativas y estructurales.	✓			✓	
Redes Neuronales	✓	✓	✓		
Máquinas de Soporte Vectorial	✓	✓	✓		
Técnicas evolutivas y Difusas	✓	✓	✓	✓	✓
Técnicas basadas en casos, densidad o distancia	✓	✓	✓		

a) Clasificación; b) Regresión c) Agrupamiento d) Reglas Asociación e) Correlaciones

Tabla 4.1: Relación entre tareas y métodos de Data Mining

manejar reglas de vecindad o funciones de distancias más complejas. En la sección 3.2 se expusieron las principales características de estos sistemas.

Todos estos métodos son fáciles de usar y eficientes para un número de ejemplos reducido. No tratan bien los atributos no significativos y al no generar modelos, no generan conocimiento comprensible. Su precisión suele ser bastante buena.

Estos son los métodos principales, aunque existen otros más específicos, dependiendo de la tarea seleccionada, y sobre todo una gran cantidad de híbridos que intentan maximizar las ventajas de cada método intentando minimizar sus puntos débiles. Aunque no es posible hacer una correlación de tareas con métodos concretos, la tabla 4.1 intenta ilustrar las principales aplicaciones de cada método.

4.6. Análisis e interpretación del conocimiento extraído

El proceso de evaluación de los patrones o conocimiento inducido es el proceso clave para determinar la eficacia de un método de minería de datos con respecto a otros métodos. Para sistemas de clasificación, la medida más directa para determinar el rendimiento del algoritmo es la estimación del error. De tal forma, que

en estos sistemas, partiendo del modelo de clasificación inducido y un conjunto de instancias de clase conocida, si dicho modelo de clasificación asigna a una instancia la clase no esperada, éste es considerado un error. La estimación del error es la proporción de errores sobre este conjunto de instancias. Existen diversos métodos para estimar la probabilidad de error de un modelo, que serán abordados en el capítulo siguiente.

Capítulo 5

Características de la Clasificación Automática

5.1. Introducción

La clasificación ocurre en un amplio rango de actividades humanas. Este amplio término cubre cualquier contexto en el cual se toman decisiones, como es el caso de la diagnosis de fallos, o se hacen previsiones basándose en la información disponible actual. Asociado a esto un procedimiento de clasificación es algún método formal que nos permite tomar repetidamente decisiones ante una nueva situación. El término clasificación también es conocido como reconocimiento de patrones, discriminación o aprendizaje supervisado.

Tiene como objetivo inducir un modelo que permita predecir, a partir de un conjunto de datos, la clase de una instancia futura [Chen *et al.*, 1996]. De manera más formal, se puede definir como:

Definición 5.1. **Clasificación.** Dado un conjunto de datos $\mathcal{D}$ (conjunto de entrenamiento), formado por un conjunto $\mathcal{A}$ de atributos, $\mathcal{A} = a_1, a_2, ..., a_m$ y un atributo C, denominado *clase*, que toma valores $c_1, c_2, ..., c_j$, el objetivo de la clasificación consiste en inducir un modelo construido mediante una función de los atributos $a_1, a_2, ..., a_m$, que permita asignar un determinado valor de la clase para las instancias futuras, previamente no conocidas, con un determinado grado de certeza.

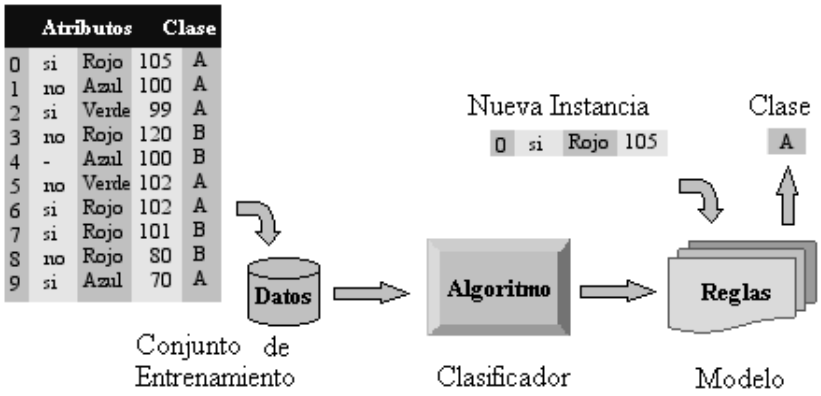

Figura 5.1: Procedimiento de clasificación

La figura 5.1 muestra, de forma gráfica, el procedimiento de clasificación. Como se puede apreciar, las instancias del conjunto de datos o conjunto de entrenamiento tienen asociada una clase. Este conjunto es analizado por el algoritmo de clasificación construyendo un modelo (en este caso representado mediante un conjunto de reglas). El modelo obtenido se utiliza para predecir la clase de nuevas instancias. Es necesario notar que existen sistemas de clasificación cuyo modelo no es ofrecido en forma de regla, como ocurre con los sistemas basados en redes neuronales.

Para llevar a cabo la clasificación se han desarrollado distintos métodos que pueden ser agrupados en varios campos de trabajo, aunque todos tienen como objetivo común construir procedimientos que sean capaces de:

- Igualar o mejorar la actividad humana consistente en tomar decisiones.
- Manejar una amplia variedad de problemas.
- Ser usados en casos prácticos, y con datos suficientes, ser generales.

En la sección 4.5 se describieron las principales técnicas de clasificación, de entre las que destacan: las técnicas estadísticas, los árboles de decisión, la similitud de instancias y las redes neuronales, por ser las más ampliamente usadas en este campo.

5.1.1. Características

Algunas de las características que nos ofrecen los clasificadores son:

- **Precisión**. La fiabilidad de la regla es representada normalmente por la proporción de clasificaciones correctas, aunque puede ser que haya errores más importantes que otros, por tanto, sería importante controlar la tasa de errores para algunas clases. Cuando el método de clasificación no es capaz de ajustarse a los datos de ninguna manera produce una tasa de errores elevada, denominándose dicho efecto *subajuste* (del ingés underfitting). Por el contrario, cuando el método de clasificación produce un número de errores de clasificación mínimos el efecto se denomina sobreajuste (del inglés overfitting). Este efecto tiene a veces efectos indeseados, pues es posible que el exceso de ajuste provoque que los *outliers* sean clasificados como pertenecientes a una clase, produciendo un modelo de clasificación erróneo. No obstante, por muy bueno que sea un método de clasificación siempre hay problemas que son resueltos mejor por otros métodos. Esta característica suele denominarse "no hay almuerzo gratis".

- **Velocidad**. En algunas circunstancias la velocidad del clasificador es una característica importante. A veces se prefiere un clasificador con menor fiabilidad si es más rápido que otro con mayor fiabilidad pero más lento.

- **Comprensibilidad**. Si es un operador humano el que debe aplicar el método de clasificación es importante que dicho método sea fácilmente entendible para evitar errores al aplicar el método. Es importante también que la persona confíe en los resultados obtenidos.

- **Tiempo de aprendizaje**. Sobre todo si estamos en entornos muy cambiantes, se necesita que el tiempo que se tarda en aprender a realizar la clasificación sea muy corto, o sea capaz de ajustar las reglas existentes a las nuevas circunstancias. El concepto de velocidad puede ser considerado como que se necesitan pocos casos nuevos para establecer la nueva clasificación. Uno de los factores que más influyen en la eficiencia del aprendizaje es el tamaño de los datos, que es función del número de ejemplos, número de atributos y la complejidad de los valores existentes. De entre ellos, el más crítico suele ser el número de atributos, sobre todo cuando dichos atributos son no relevantes. Este problema suele denominarse *la maldición de la dimensiona-*

lidad, y hay técnicas como por ejemplo las basadas en distancia o similitud que son muy sensibles a dicho problema.

Existen una serie de algoritmos denominados *algoritmos anytime* que producen mejores resultados cuanto más tiempo se le de al algoritmo. Dichos algoritmos pueden ser parados en cualquier momento y ofrecerán la mejor solución encontrada hasta ese instante.

5.2. Estructura General de los Problemas de Clasificación

Hay tres componentes esenciales en un problema de clasificación:

1. La frecuencia relativa con que las clases aparecen en la población de estudio, expresada formalmente como la distribución de probabilidad principal.
2. Un criterio implícito o explícito para separar las clases.
3. El coste asociado a realizar una clasificación incorrecta.

La mayoría de las técnicas implícitamente confunden casos y, por ejemplo, producen unas reglas de clasificación que se derivan de la distribución especifica del conjunto de aprendizaje y no pueden ser fácilmente adaptadas cuando se produce un cambio en la frecuencia de las clases. En cambio, en teoría, cada uno de esos casos puede ser individualmente estudiado y de esta forma combinar los resultados en una sola regla de clasificación. A continuación describiremos dicho desarrollo.

5.2.1. Probabilidad a priori y la regla por defecto

Necesitaremos introducir algunas notaciones previas. Sean las clases $A_i, i = 1, ...q$ y sea la probabilidad pi de cada clase $p_i = p(A_i)$. Siempre es posible encontrar la regla por defecto, es decir, clasificar una nueva observación como A_k independientemente de los atributos del ejemplo. Esta regla por defecto puede ser incluso adoptada en la práctica si el coste de adquisición de los datos es excesivamente alto. La regla por defecto cuenta tan sólo con el conocimiento de la probabilidad a priori, y claramente la regla de decisión que tiene la probabilidad más grande es la clase más frecuente.

5.2.2. Separación de clases

Supongamos que podemos hacer observaciones individuales y que conocemos que la distribución de probabilidad de cada clase A_i es $P(x|A_i)$. Entonces para dos clases cualesquiera la razón de probabilidad $P(x|A_i)/P(x|A_i)$ es la forma óptima teórica de diferencias las clases basándose en los datos.

Existen una serie de funciones heurísticas que permiten determinar, de forma aproximada, la complejidad geométrica de un problema. Estas funciones determinan el grado de *separabilidad*, considerando una medida de distancia.

Existe **separabilidad lineal** [Minsky and Papert, 1969], si existe una función lineal (una recta, un plano o un hiperplano) que separe los dos grupos o clases nítidamente.

La **separabilidad geométrica** [Thornton, 1997] se define como:

$$SG(f) = \frac{\sum_{i=1}^{n} eq(f(e_i), f(nn(e_i)))}{n} \tag{5.1}$$

donde f es la función definida por los datos, n es el número de ejemplos, nn es el vecino más cercano (según la medida de distancia), y $eq(a, b) = 1$ si $a = b$ y 0 en caso contrario. Esta función ajusta mejor a otro tipo de patrones, aunque no sean lineales, pero depende mucho del número de ejemplos.

La **separabilidad geométrica radial** es una mejora de la anterior, y se define como:

$$SGR(f) = \frac{\sum_{i=1}^{n} \frac{\sum_{e_j:dist(e_i,e_j)<r} eq(f(e_i),f(e_j))}{|e_j:dist(e_i,e_j)<r|}}{n} \tag{5.2}$$

donde *dist* representa la función distancia entre dos ejemplos, y r es el radio de elementos de la misma clase. Precisamente, establecer el valor adecuado para el radio es muy importante. Dicho valor puede ser estimado a partir de la densidad de los ejemplos.

5.2.3. Coste de errar la clasificación

Supongamos que el costo asociado a clasificar un objeto de la clase A_i como perteneciente a la clase A_j es $c(i, j)$. La decisión debería basarse en el principio de que el coste total de realizar una clasificación errónea debería ser minimizado.

Para una nueva observación esto significa minimizar el coste esperado de una clasificación errónea.

Primero consideremos el coste de aplicar la regla por defecto: asociar cualquier observación nueva a la clase A_d, asumiendo que d es la etiqueta de la clase. Cuando se toma la decisión A_d para todos los nuevos casos, para los casos de la clase A_i tenemos un coste $c(i, d)$ con una probabilidad p_i. De esta forma el coste esperado C_d de tomar la decisión A_d es:

$$c_d = \sum_i p_i c(i, d) \tag{5.3}$$

En la práctica el coste de errar en la clasificación es muy difícil de obtener, incluso en casos en los que es muy claro la desigualdad existente entre las recompensas o penalizaciones por tomar una decisión correcta o incorrecta. La manera más habitual de expresar dichos costes es mediante la **matriz de costes**. Esta matriz representa los costes de todas las posibles combinaciones que se pueden dar entre la clase predicha y la real. Por tanto, para un problema de C clases, la matriz será de tamaño CxC. Cada celda representa el coste de errar, según la clasificación correcta y la predicha.

5.3. Técnicas para mejorar la precisión

En aquellos problemas donde la separabilidad de las clases sea muy baja, la precisión obtenida por los métodos de clasificación podrán variar en función del método, pero serán en general peor que para clases más fácilmente separables. En estos casos, podemos usar una serie de métodos que nos permiten aumentar la precisión:

- Aumento de la dimensionalidad y/o creación de atributos. Generalmente al aumentar la dimensionalidad podemos definir separabilidad lineal o más sencilla al tener más dimensiones con las que definir el separador. Este es el caso de las máquinas de soporte vectorial. No obstante, existe también la posibilidad de combinar atributos para aprender conceptos que no pueden aprenderse con los atributos originales.

- Localidad. Existen problemas que globalmente son muy poco separables, pero que sin embargo pueden aproximarse mejor a un nivel más local. El problema que plantea esta opción es que podemos caer en el sobreajuste, al tratar un único problema global como la unión de una serie de problemas locales.

- Combinación de modelos y métodos. Consiste en crear modelos formados por la combinación de varios modelos. Este caso será tratado en las secciones 5.5 y 5.6.

5.4. Métodos para la evaluación de resultados

Es de sobra conocido que si estimamos los porcentajes de error con el mismo conjunto de datos con los que se ha realizado la clasificación los resultados no son muy fiables. Por una parte si permitimos un árbol de clasificación lo suficientemente complejo, es decir, con el suficiente número de niveles y hojas, posiblemente podamos obtener una tasa de acierto del 100 % sobre los mismos datos usados para generar las reglas.

En la práctica estas estructuras tan complejas no clasifican bien sobre datos distintos a los del entrenamiento, y este es uno de los principales problemas de 'sobreajustar' los datos. El fenómeno de sobreajuste es más preocupante aún cuando, sobre los datos de entrenamiento, es imposible obtener un porcentaje de acierto del 100 % debido a la existencia de casos conflictivos, aunque también es preocupante aunque esto no ocurra. Como regla general deberemos usar estructuras de clasificación simples para casos en los que los datos de entrenamiento contengan casos imposibles de clasificar correctamente, y dejar las estructuras más complejas para casos en los que la clasificación de los datos de entrenamiento es posible hacerla con el 100 % de acierto.

Entrenamiento y Test

La idea básica es obtener unas reglas de clasificación a partir de un conjunto de datos de entrenamiento, y conocer el porcentaje de errores que se obtienen al clasificar un conjunto de datos distintos a los empleados para generar las reglas, y sobre los que no se conoce la clasificación correcta. Para realizar esto vamos a testar

las reglas con un conjunto de datos distintos a los de entrenamientos y sobre los que si se conoce la clasificación. Para poder hacer esto vamos a separar del conjunto de entrenamiento un subconjunto aleatorio (normalmente el 20 o 30 % de casos) de casos que no serán usados en el entrenamiento y nos servirán posteriormente para realizar los tests. Hay una perdida de eficiencia al no entrenar la clasificación con todos los casos disponibles, pero en bases de datos grandes tampoco supone mucho problema. Podremos usar este procedimiento cuando el número de casos sea mucho mayor de 1000 y podamos usar un conjunto de test de aproximadamente 1000 casos.

Validación cruzada (Cross-validation)

Para casos en los que el tamaño de los casos sea moderado el método empleado es el de validación cruzada [Lachenbruch and Mickey, 1968]. El método consiste en dividir los datos disponibles en subconjuntos. Cada subconjunto se predice con las reglas de clasificación obtenidas de los (m-1) subconjuntos restantes. De esta manera el error se calcula de forma imparcial y eficiente. Finalmente se utilizan todos los datos para obtener las reglas definitivas. [Stone, 1974] describe los métodos de validación cruzada para dar estimaciones imparciales de la proporción del error.

Una dificultad para usar el método de validación cruzada en métodos que requieren altos grados de tiempo de computación, como pueden ser las redes neuronales, es la repetición del entrenamiento para cada subconjunto, lo que puede producir tiempos de computación excesivos.

Validación por secuencia (Bootstrap)

La principal objeción impuesta al método de validación cruzada es que produce una estimación del error demasiado dispersa, ya que los intervalos de confianza para la tasa de error real son demasiado amplios. El procedimiento de validación cruzada produce un intervalo de confianza mucho más estrecho, pero el precio a pagar es que las tasas de error estimadas están optimizadas. Normalmente es preferible el uso de validación en secuencia cuando el tamaño de la base de datos es pequeño, mientras que para bases de datos más grandes es preferible el método

de validación cruzada. El método consiste en analizar repetidamente subconjuntos de datos. Cada subconjunto se toma al azar con reemplazo a partir del original.

[Efron, 1983] proporciona las siguientes propiedades de la validación por secuencia como un estimador de la tasa de error:

- Tomando un número de subconjuntos muy grande la varianza estadística en la media de la tasa de error es pequeña, y para ejemplos de tamaño pequeño esto significa que la validación por secuencia tendrá mucha menor varianza estadísticamente que la estimación por validación cruzada.
- La validación por secuencia y la validación cruzada son muy parecidas para ejemplos de tamaño grande y el porcentaje entre las dos estimaciones tiende a 1 cuando el tamaño del ejemplo tiende a infinito.
- El tamaño efectivo del ejemplo viene determinado por el número en el grupo de clasificación más pequeño. En [Efron, 1983] se cita un ejemplo médico con 155 casos, pero el interés primario se centra en los 33 pacientes que mueren. Así pues el tamaño efectivo del ejemplo es de 33.

Matriz de confusión

Una matriz de confusión C es una matriz de $m \times m$ enteros que contienen el rendimiento de un clasificador φ, aplicado a los ejemplos de una base de datos. La entrada $C(i,j)$ cuenta el número de realmente pertenecen a la clase i, pero que fueron clasificados por φ como pertenecientes a la clase j.

$$C(i,j) = \{e \in D | D(e) = i, D_\varphi(e) = j\} \tag{5.4}$$

Obviamente es deseable que los elementos de la diagonal principal tengan los valores tan grandes como sea posible. Las probabilidades se estiman fácilmente a partir de la matriz C dividiendo cada entrada por la suma de las filas o columnas donde aparece la entrada.

$$Pr[D(e) = i | D_\varphi(e) = j] = \frac{C(i,j)}{\sum_{i=1}^{m} C(i,j)} \tag{5.5}$$

$$Pr[D_\varphi(e) = j | D(e) = i] = \frac{C(i,j)}{\sum_{j=1}^{m} C(i,j)} \tag{5.6}$$

	Real	
Estimado	TPR	FPR
	FNR	TNR

Tabla 5.1: Matriz de Confusión Normalizada

$$Pr[D_{\varphi}(e) = D(e)] = \frac{\sum_{i=1}^{m} C(i,j)}{\sum_{j=1}^{m}\sum_{i=1}^{m} C(i,j)} \tag{5.7}$$

La proporción de ejemplos correctamente clasificados se denomina *exactitud de predicción* (ecuación 5.7) y es sin duda la medida de rendimiento más popular en el área de aprendizaje automático.

Análisis ROC

Todas las técnicas anteriores evalúan la clasificación basándose en el porcentaje de error cometido, sin diferenciar en el tipo de error cometido, sin embargo, sabemos que los diferentes errores tienen costes muy dispares. Si se dispone de la matriz de costes, se podría adaptar el aprendizaje al contexto de coste determinado. Desgraciadamente, la matriz de costes no siempre está disponible. En estos casos, lo que se suele hacer es aprender un conjunto de clasificadores y seleccionar el que mejor se comporte para un contexto de coste determinado a posteriori.

El análisis ROC (del inglés Receiver Operating Characteristic) provee herramientas para seleccionar el conjunto de clasificadores que tienen un comportamiento óptimo en general, y evaluar los clasificadores de manera más independiente y completa que la precisión del error.

Se utiliza para problemas de dos clases (clase positiva y negativa). Para este tipo de problemas utiliza un matriz de confusión normalizada, del tipo:

Donde TPR es el porcentaje de positivos: $TPR = TP(TP/FN)$, FNR es el porcentaje de falsos negativos $FNR = FN/(TP + FN)$, FPR es el porcentaje de falsos positivos $FPR = FP/(FP + TN)$ y TNR es el porcentaje de negativos $TNR = TN/(FP + TN)$. Siendo TP el número de positivos, TN el número de negativos, FP el número de falsos positivos y FN el número de falsos negativos. Lógicamente, $TPR = 1 - FNR$ y $FPR = 1 - TNR$. Esto permite que con los ratios TPR y FPR se pueda construir el denominado *espacio ROC*, limitado por

los puntos $(0, 0)$ y $(1, 1)$. Habitualmente se usa el valor FPR para el eje de las x y el valor de TPR para el eje de las y.

Representar un clasificador de esta forma en el espacio ROC permite que dado un conjunto de clasificadores, se pueden descartar los clasificadores que no caigan dentro de la superficie convexa formada por todos los clasificadores, junto con los puntos (0,1), (1,0) y (0,0). Por tanto, el mejor sistema de aprendizaje será aquel que produzca el conjunto de clasificadores con mayor area bajo la superficie convexa. Este área se denomina AUC (del inglés Area Under the ROC Curve). La figura 5.2 muestra un ejemplo de diagrama ROC. Los clasificadores B y E pueden ser descartados, según el análisis ROC.

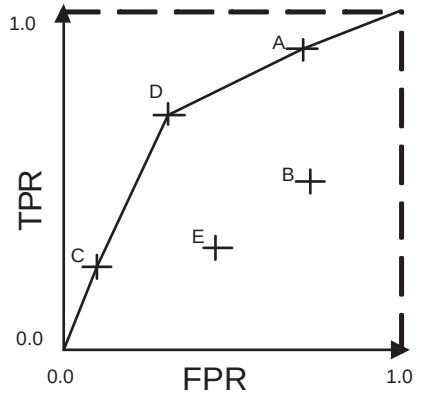

Figura 5.2: Ejemplo de diagrama ROC

La principal limitación del análisis ROC es que sólo es fácilmente aplicable a problemas de dos clases.

5.5. Construcción de multiclasificadores

En los últimos años, ha surgido un creciente interés por la combinación de clasificadores con el objeto de mejorar la precisión. Diettrich [Dietterich, 2000a], establece una taxonomía para la construcción de conjuntos de clasificadores:

- Manipulación de los datos de entrenamiento. Son métodos construyen varios clasificadores al aplicar el mismo algoritmo de aprendizaje a un conjunto diferente de datos de entrenamiento. Lo importante es establecer un mecanismo adecuado para seleccionar los subconjuntos de entrenamiento.

- Manipulación de los atributos de entrada. Consiste en alterar el conjunto de atributos de entrada del algoritmo de aprendizaje.
- Manipulación de los datos de salida. Modifican las clases de los ejemplos del conjunto de entrenamiento.
- Métodos aleatorios. Introducen componentes aleatorios en los procesos de aprendizaje.

Los métodos para construcción de multiclasificadores más habituales son:

Bagging. El termino bagging deriva de bootstrap aggregation, que es un mecanismo para generar subconjuntos de entrenamiento seleccionando aleatoriamente y con reemplazamiento (puede haber ejemplos repetidos) una muestra de m ejemplos de entrenamiento del conjunto original de entrenamiento, formado por m ejemplos. Podría esperarse que si usamos siempre el mismo algoritmo de aprendizaje, el modelo obtenido fuese similar en todos los casos, pero hay algoritmos de aprendizaje que son muy sensibles al conjunto de entrenamiento, como por ejemplo los árboles de decisión. Puesto que obtenemos un conjunto de clasificadores, la nueva predicción se efectúa por votación mayoritaria.

Boosting. El mecanismo se basa en la asignación de un peso a cada ejemplo del conjunto de entrenamiento. En cada iteración, el boosting aprende un modelo que minimiza la suma de los pesos de los ejemplos clasificados erróneamente. Los errores de cada iteración se usan para actualizar los pesos de los ejemplos del conjunto de entrenamiento, de forma que aumenta el peso de los ejemplos fallidos y decrementa el peso de los acertados. Existen muchas variantes del algoritmo básico de boosting, siendo el más popular AdaBoost [Freund and Schapire, 1996]. A diferencia de bagging, el algoritmo de boosting no realiza siempre las k iteraciones dado que considera un criterio de parada de acuerdo con el error obtenido.

Randomization. Es un método multiclasificador basado en introducir componentes aleatorias en las decisiones internas del algoritmo de construcción de árboles de decisión. La idea consiste en seleccionar de forma aleatoria la

partición donde continuar la construcción del árbol entre las veinte mejores particiones. Así pues, se obtiene un árbol distinto en cada iteración.

En [Dietterich, 2000b] se encuentra una comparativa experimental de los métodos comentados.

5.6. Métodos híbridos

Son métodos de aprendizaje formados por combinación de otros métodos de aprendizaje. Buscan combinar varios métodos de aprendizaje para aunar las ventajas de cada uno. Los más relevantes son:

Stacking. Se basa en combinar los modelos generados por diferentes algoritmos de aprendizaje [Wolpert, 1992]. Puesto que cada método tiene una precisión distinta, la votación mayoritaria entre los resultados de los métodos no refleja el resultado óptimo. Una solución es introducir una fase de meta-aprendizaje en el modelo combinado. Con esto intentamos aprender la mejor manera de combinar los modelos base. En el meta-aprendizaje se recibe como entrada las predicciones de los modelos base junto con la clase del problema a aprender. La figura 5.3 muestra el esquema general del stacking [Hernández Orallo *et al.*, 2004].

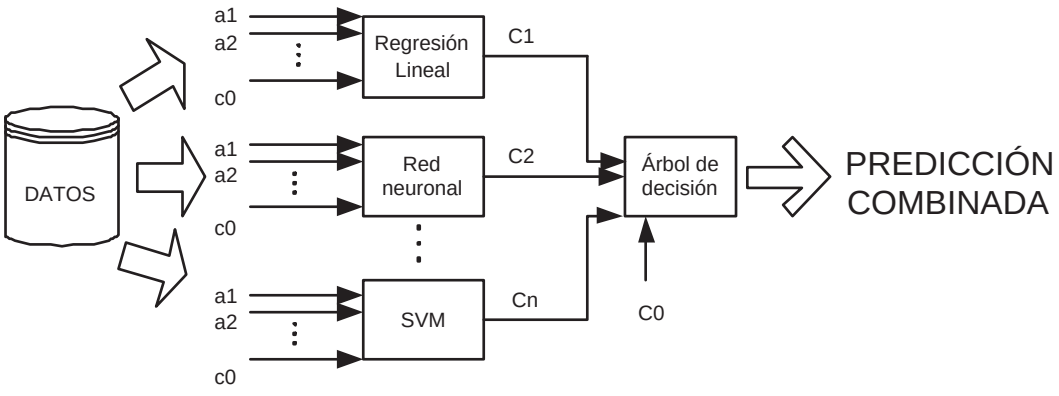

Figura 5.3: Esquema general de Stacking

Cascading. Permite mejorar las características de los árboles de decisión mediante la incorporación de nuevas particiones basadas en otras técnicas de

aprendizaje. La idea consiste en usar otros métodos de aprendizaje sobre alguno de los atributos para crear nuevos atributos artificiales. Estos nuevos atributos son usados también por el árbol de decisión para generar el modelo. La figura 5.4 muestra el esquema general del cascading [Hernández Orallo *et al.*, 2004].

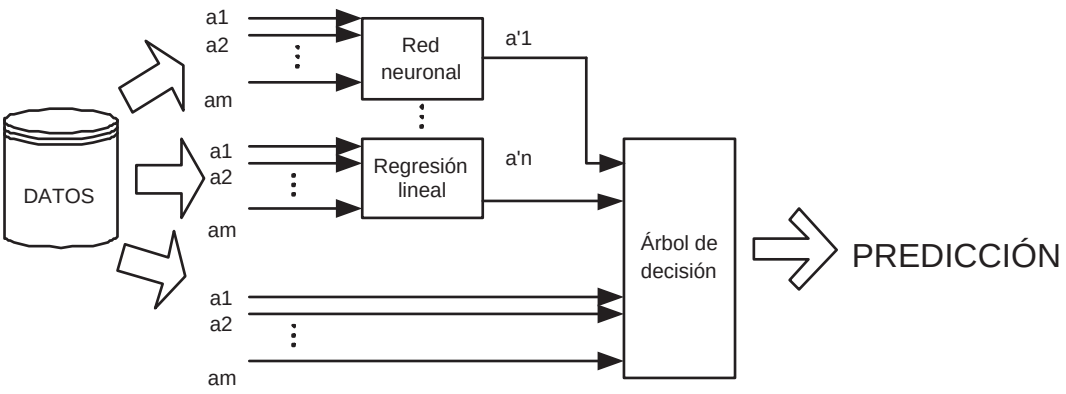

Figura 5.4: Esquema general de Cascading

Capítulo 6

Diagnosis Mediante Aprendizaje de Modelos Proposicionales

6.1. Introducción

La detección de fallos, e identificación de los componentes defectuosos, son tareas muy importantes desde el punto de vista estratégico para las empresas, debido a las demandas económicas y medioambientales que deben cumplir para ser competitivas en los mercados. Estas tareas se vuelven más complejas cuando los sistemas a diagnosticar son sistemas dinámicos.

Estos sistemas son muy difíciles de diagnosticar debido al gran número de componentes que lo integran, la poca cantidad de variables observables de que disponen, y la presencia habitual de lazos de control que habitualmente ocultan la presencia de los fallos y hacen más difícil su identificación.

Como ya se ha citado en el capítulo 3, existe una amplia variedad de técnicas, basadas en la Inteligencia Artificial, que se han venido aplicando al campo de la diagnosis desde hace más de una década [Feng, 1992]: lógica borrosa [Khoo *et al.*, 2000; Garcia *et al.*, 2000], redes neuronales [Karpenko and Sepehri, 2002; Seker *et al.*, 2003], razonamiento basado en casos [Bregon *et al.*, 2006; Pous *et al.*, 2003], o conjuntos robustos [Tay and Shen, 2003], entre otras. En [Isermann, 2006] podemos encontrar una revisión de técnicas de diagnosis de fallos que usan métodos de clasificación. Muchos de dichos métodos se encuentran dentro de la *diagnosis basada en modelos* (Sección 3.3), que basan su diagnóstico en la evaluación de

la discrepancia entre las lecturas de los sensores y las salidas de un modelo del sistema.

Dentro de las técnicas que forman la Inteligencia Artificial, la minería de datos resuelve los problemas a través del análisis de los datos existentes en bases de datos. Puede considerarse como el proceso automático de descubrir patrones en los datos. Uno de los campos de la minería de datos es el aprendizaje automático, que puede considerarse como la habilidad de un ordenador de generar nuevo conocimiento en base a experiencias anteriores. Las técnicas de aprendizaje automático se vienen aplicando a la diagnosis desde principios de los noventa [Feng, 1992] hasta nuestros días [Roverso, 2003; Zhao *et al.*, 2005; Yahui *et al.*, 2007; Widodo and Yang, 2007].

Tal como se expuso en el capítulo 2 el **objetivo** de la presente tesis es conseguir diagnosticar sistemas dinámicos continuos en base a las experiencias que previamente se tienen almacenadas de ellos. Para lograr dicho objetivo se plantea una metodología que se adapta a las fases del proceso de $\mathcal{KDD}$, especificando los tratamientos a realizar en sus diferentes fases.

La metodología propuesta comienza con la recogida de las medidas ofrecidas por los sensores disponibles del sistema. Estas medidas serán tomadas y almacenadas tanto para el sistema funcionando correctamente, como para cada una de las situaciones de fallos en las que el sistema necesite ser diagnosticado. Así pues, cada una de dichas situaciones deberá ser provocada *off-line*, con anterioridad a la fase de diagnóstico. Normalmente, dicha información deberá ser tratada en varios sentidos, desde la limpieza y filtrado de los datos tomados, hasta la reducción de la información no relevante o la adición de atributos no presentes en los datos originales, pero que sí serán significativos para el proceso de diagnóstico.

Habrá ocasiones, en las que el sistema no pueda ser manipulado para producir el fallo que se quiere diagnosticar. Estos serán, por ejemplo, los casos en los que el fallo provocaría la destrucción del sistema, o se necesitaría parar la producción para generarlos. Para este tipo de fallos, se puede recurrir a un modelo del sistema que nos proporcione la información que el sistema propiamente dicho no puede darnos. Dicha información se añadirá a la recogida propiamente del sistema real.

Una vez que la información recogida ha sido tratada y almacenada, se aplicará sobre dichos datos alguna herramienta de clasificación que nos permita inferir conocimiento de dicha información. Ese conocimiento vendrá dado en forma de

árbol de decisión, que será capaz de catalogar los comportamientos del sistema en base a los datos recogidos. Todo este proceso es desarrollado *off-line*, y el producto final será el árbol de decisión que nos servirá en la fase de diagnosis.

La fase de diagnosis se desarrollará *on-line*, con el sistema funcionando, y mientras está siendo monitorizado. Las lecturas de los sensores serán evaluadas con el árbol de decisión obtenido para alcanzar un diagnóstico.

6.2. Metodología general

En esta sección se va a exponer una metodología general para realizar diagnosis de sistemas dinámicos. Dicha metodología se basa en la adaptación de las fases del $\mathcal{KDD}$ a la diagnosis de sistemas dinámicos.

Previamente se necesitará indicar las limitaciones de la metodología y las definiciones y notación necesarias para exponer la metodología.

6.2.1. Condiciones supuestas en la metodología

Es necesario considerar algunas condiciones que deben cumplirse a la hora de aplicar la presente metodología:

- El sistema a diagnosticar deberá estar alternando entre estados transitorios y estacionarios. Es decir, el sistema evoluciona desde un punto de operación a otro cada cierto tiempo.
- El número de estados transitorios del sistema es finito y conocido, y se produce siempre entre dos puntos de operación conocidos.
- Cuando el sistema comienza un estado transitorio, de existir un fallo, éste está presente totalmente desde el principio del transitorio y no evoluciona hasta llegar al estacionario.
- Los fallos del sistema sólo podrán ser detectados durante los estados transitorios, independientemente de cuando dichos fallos ocurran.
- Todos las situaciones de fallos deben haber sido reproducidas antes de poder ser diagnosticadas.

6.2.2. Definiciones y notación

Para poder clarificar los pasos a seguir en la metodología propuesta, son necesarias las siguientes definiciones:

Los sistemas dinámicos, que trataremos en la presente tesis, tendrían la siguiente representación simbólica:

$$\Phi(dX/dt, X, P, t) = 0, \quad X \in I\!R^n, \quad P \in I\!R^m \quad n, m \in I\!N \tag{6.1}$$

donde X representa el vector de estado, P es el vector de parámetros y t representa el tiempo. En algunos sistemas la relación Φ es imprecisa o desconocida. El objetivo de la presente metodología es inducir dicha información partiendo del comportamiento del sistema. Para aclarar dicho propósito, se necesitarán las siguientes definiciones:

Definición 6.1. **Trayectoria**. Una *trayectoria* representa la evolución en el tiempo de una variable observable del sistema dinámico Φ, desde un estado inicial $x_0 \in I\!R^n$ hasta un estado final $x_f \in I\!R^n$. Los elementos de una *trayectoria* pueden ser definidas por la función:

$$\Psi : \Gamma \rightarrow X \tag{6.2}$$

donde $X \in I\!R^n$ y Γ es un conjunto finito de k instantes de tiempo que tienen los siguientes valores:

$$\Gamma = \{t_0, t_1 = t_0 + \Delta t, t_2 = t_0 + 2\Delta t, \ldots, t_{k-1} = t_0 + (k-1)\Delta t\} \tag{6.3}$$

donde Δt viene dado por la frecuencia de muestreo de la variable observable. Cada *trayectoria* se representa como j. Un conjunto finito de *trayectorias* es denotada como J, donde $J = \bigcup_{i=1}^{n} j_i$, siendo n el número de *trayectorias*.

Definición 6.2. **Trayectoria etiquetada**. Desde el punto de vista del correcto funcionamiento del sistema, sus diferentes comportamientos serán denotados mediante un conjunto finito de etiquetas B. El conjunto B contendrá una etiqueta que denotará la ausencia de fallos, y una etiqueta por cada uno de los fallos que pueden ser provocados en el sistema.

Una *trayectoria etiquetada* es un par $\langle j, b \rangle$, donde el primer elemento es una *trayectoria* $j \in J$, y el segundo es una etiqueta $b \in B$ que representa el comportamiento que tenía el sistema cuando se obtuvo dicha *trayectoria*.

El conjunto de todas las *trayectorias etiquetadas* de un sistema es almacenado en una *base de datos etiquetada* denotada por L. Nótese que existirá una *base de datos etiquetada* por cada una de las variables observables del sistema.

Definición 6.3. **Trayectoria transformada**. Las *trayectorias etiquetadas* poseen la información de los sensores del sistema tal y como fueron recogidos. Cualquier manipulación que aumente, disminuya o altere dicha información producirá una nueva *trayectoria* a la que denominaremos *trayectoria transformada*. El conjunto de todas las *trayectorias transformadas* del sistema es almacenado en una *base de datos transformada*, a la que denotaremos como T.

6.2.3. Fases de la metodología

La metodología general consta de dos fases claramente diferenciadas:

1. La primera fase se lleva a cabo of-line (fig. 6.1). En esta fase el objetivo es obtener un modelo proposicional que caracterice el comportamiento del sistema para los diferentes fallos que quieren ser diagnosticados.

   La primera tarea de la fase *off-line* es seleccionar el conjunto de fallos que pretenden ser diagnosticados y los estados transitorios en los cuales podrán ser detectados.

   Puesto que la diagnosis deberá ser desarrollada únicamente en los estados transitorios, todo el proceso deberá ser realizado para cada uno de los transitorios en los que se desee diagnosticar el sistema. Se obtendrá por tanto un modelo distinto para cada uno de los transitorios a diagnosticar.

   La obtención del modelo proposicional resultante de la fase *off-line* se realizará adaptando las fases del $\mathcal{KDD}$ al problema de diagnóstico planteado. Para ello, se comienza recogiendo una colección de *trayectorias* pertenecientes a cada comportamiento a diagnosticar, para crear con ellas las *bases de datos etiquetadas*. Dichas *bases de datos etiquetadas* serán limpiadas y tratadas

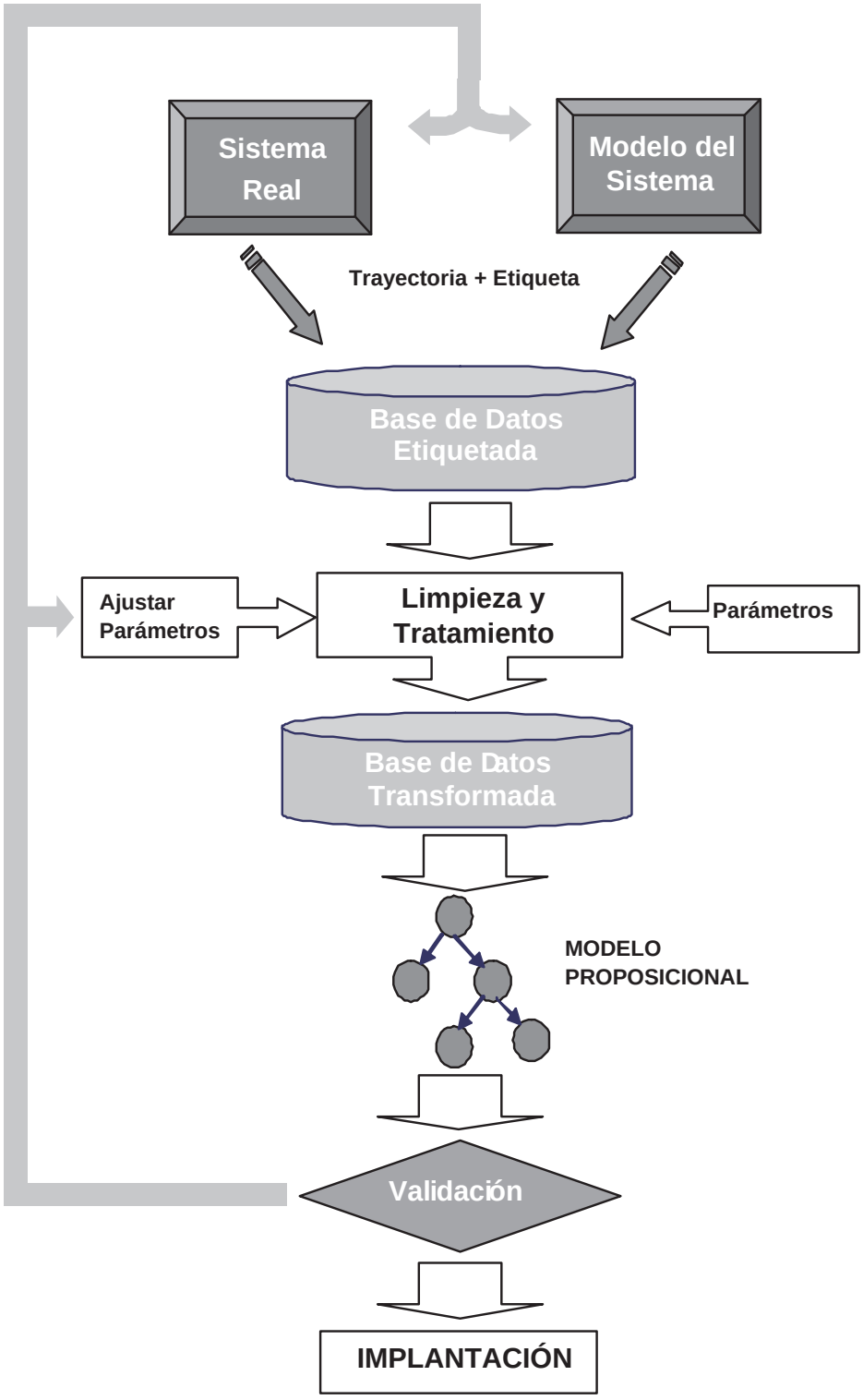

Figura 6.1: Fase *Off-line* de la metodología general

para conseguir un conjunto de información representativa del conocimiento que se quiere extraer. A la *base de datos transformada*, resultante del proceso anterior, se aplicará posteriormente una herramienta de clasificación, que devolverá el modelo proposicional resultante. Si la validación de dicho modelo es positiva se procederá a la fase de implantación, donde la diagnosis será desarrollada *on-line*. De no ser positiva la evaluación, se deberá recoger más información o ajustar los parámetros del tratamiento de las *trayectorias* con el fin de obtener un modelo más preciso.

En las siguientes secciones se exponen los diferentes pasos que se deben aplicar para la consecución del modelo.

2. La segunda fase (fig. 6.2) es la fase de diagnosis propiamente dicha. Se desarrolla *on-line*, mientras el sistema está siendo monitorizado. En esta fase, las medidas de los sensores del sistema son evaluadas mediante el modelo obtenido en la fase anterior para conseguir el diagnóstico.

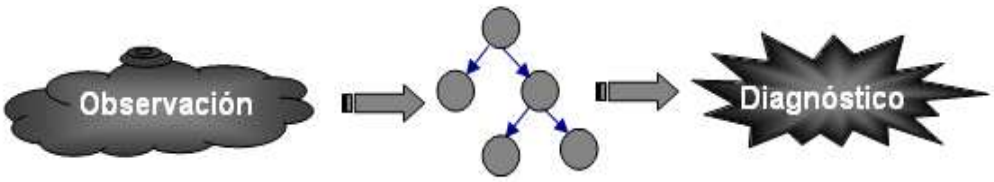

Figura 6.2: Fase *on-line* de la metodología general

La fase *on-line* consiste en evaluar una nueva observación, recogida de los sensores del sistema monitorizado, mediante el modelo conseguido en la fase *off-line*. Antes de poder evaluar la observación, necesitamos seleccionar el modelo correspondiente al transitorio en el que se encuentra el sistema. Por tanto, habrá que identificar el comienzo del transitorio, el punto de operación en que se encuentra el sistema y el nuevo punto de operación al que se quiere llevar. Dicha identificación será realizada mediante la monitorización del sistema y el cambio de consigna realizado para el cambio de punto de operación. Una vez identificado el transitorio, se seleccionará el modelo generado para dicho transitorio.

Para poder evaluar la nueva observación, será necesario que los tratamientos efectuados sobre las *trayectorias* con las que se ha generado el modelo sean

también efectuadas sobre la nueva observación. De esta forma, se generarán para la nueva observación aquellos atributos no presentes en las *trayectorias* originales.

La evaluación de la observación, devolverá una etiqueta del conjunto B, que asociará la nueva observación a un comportamiento del sistema. Dicha etiqueta corresponderá con el diagnóstico del sistema.

El proceso de evaluar el modelo deberá ser muy rápido, no necesitando gran capacidad de cómputo ni de almacenamiento. Esto permite que la diagnosis se pueda llevar a cabo en tiempo real, incluso para sistemas cuya frecuencia de muestreo es muy alta y que evolucionan muy rápidamente. Las pocas necesidades computacionales y de almacenamiento, permiten además que los costes en el equipamiento del sistema de diagnóstico sean muy contenidos.

6.3. Creación de la *base de datos de trayectorias*

Partiendo del sistema a diagnosticar, la primera tarea que se debe realizar es seleccionar los estados transitorios que pretenden ser diagnosticados. Recordemos que la metodología se plantea únicamente para la diagnosis de sistemas dinámicos continuos en los estados transitorios, es decir, cuando el sistema necesita cambiar de un estado a otro. Es habitual que dichos cambios se deban a cambios en la consigna o en algún parámetro que regula el funcionamiento del sistema.

Por tanto se deberá realizar una selección de los transitorios a diagnosticar, precisando el evento que provoca el cambio de estado. En la figura 6.3 se representan posibles transitorios para una determinada variable observable de un mismo sistema.

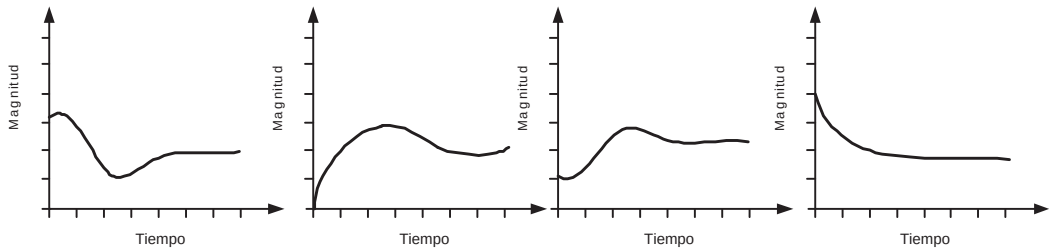

Figura 6.3: Distintos transitorios de un sistema

Una vez seleccionados los transitorios, el siguiente paso será seleccionar los distintos fallos que se pretenden diagnosticar en cada uno de ellos. Los fallos pueden ser derivados de la misma causa para los diferentes transitorios, o pueden ser fallos diferentes para cada uno de ellos. Además, dependiendo de los componentes del sistema que fallen se podrán diagnosticar:

- Fallos simples: Cuando sólo falla un componente del sistema.
- Fallos múltiples: Cuando el fallo se deriva de más de un componente del sistema que falla al mismo tiempo.

Puesto que los fallos que se pretenden diagnosticar deberán ser reproducidos de alguna forma, podemos realizar la diagnosis tanto de fallos simples como de fallos múltiples. A cada uno de los fallos que se quiera diagnosticar, independientemente de su tipo se le asignará una etiqueta, que será el identificador de dicho fallo durante todo el proceso.

Finalmente, para poder realizar la recogida de la información se deberán seleccionar las variables observables del sistema con las que se pretende realizar el diagnóstico. A priori, puede resultar interesante capturar todas las variables observables del sistema, pero dependiendo de cada situación se seleccionarán todas o algún subconjunto de ellas. La figura 6.4 representa diferentes variables observables de un sistema. Las variables seleccionadas serán aquellas que deberán ser monitorizadas durante la fase de diagnóstico.

Así pues, antes de comenzar la recogida de la información se necesitan fijar:

- Transitorios durante los cuales el sistema va a ser diagnosticado.
- Fallos que pretenden ser diagnosticados.
- Variables observables del sistema que van a ser monitorizadas.

Una vez fijados los parámetros, se necesita conseguir una colección de *trayectorias* del sistema, tanto para el funcionamiento exento de fallos como para cada una de las situaciones de fallo. Habitualmente, obtener las *trayectorias* del sistema cuando este funciona correctamente no plantea ningún problema, pero la obtención de las *trayectorias* de fallo es otra cosa. Con este propósito, se plantean dos alternativas:

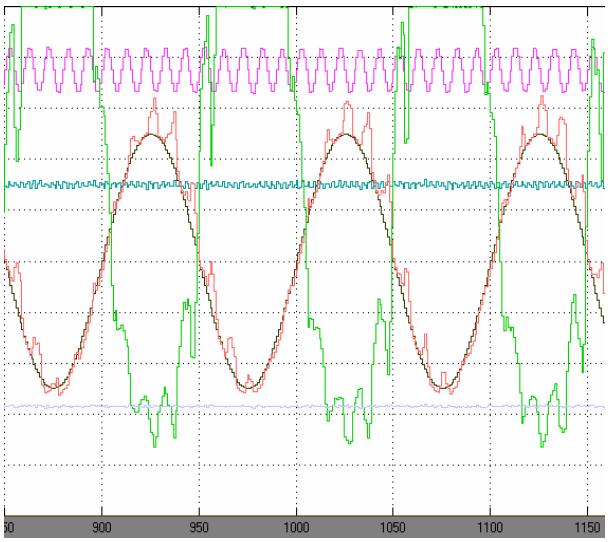

Figura 6.4: Distintas variables observables de un sistema

- Cuando es posible manipular el sistema para provocar el fallo sin necesidad de dañar el sistema, se provoca el fallo y se recogen las *trayectorias*. Por supuesto, si el sistema forma parte de una línea de producción o es un subsistema de otro mayor, será necesario aislarlo para no afectar el entorno. Estos son, por ejemplo, los fallos que pueden ocurrir debido a la desconexión accidental de algún componente. También es posible provocar los fallos cuando el sistema a diagnosticar es un producto que va a ser producido en serie en grandes cantidades. En este caso, los fallos pueden ser provocados en algunos prototipos aunque dichos fallos destruyan el prototipo, siempre y cuando los costes lo permitan. De esta forma se obtendrían las *trayectorias* de fallo. El diagnosticador obtenido en este caso podría servir para todos los productos de la serie.

- Cuando el fallo no puede ser provocado en el sistema real (porque el sistema resultaría dañado, o sea imposible desconectarlo para realizar los ensayos), se deberá generar un modelo del sistema. Dicho modelo deberá representar fidedignamente al sistema y ofrecer como salida los valores de todos los sensores del mismo. En este caso, las *trayectorias* de fallo no se obtendrían del sistema, sino de las simulaciones de los diferentes fallos en el modelo.

Estas dos opciones no son excluyentes, y ambas pueden ser usadas a la vez.

Finalmente, se añadirá a cada *trayectoria* recogida la etiqueta representativa con el modo de fallo en que fue obtenida. De esta forma, cada *trayectoria* se ha transformado en una *trayectoria etiquetada*. Todas las *trayectorias etiquetadas* serán almacenadas en las *bases de datos etiquetadas*, que será el resultado final de este paso.

El número de *trayectorias* recogidas por cada situación de fallo, o la ausencia del mismo, deberá ser fijado a priori. Obviamente, mientras más *trayectorias* se obtengan, se tendrá más información para inducir el modelo proposicional, aunque puede ocurrir que a partir de un determinado número de *trayectorias*, las *trayectorias* adicionales no aporten información significativa.

Además, se deberá tener en cuenta los factores que pueden alterar la evolución de una *trayectoria* dentro de un determinado comportamiento. Por ejemplo, la temperatura ambiental puede alterar la evolución de una variable del sistema dentro de un comportamiento específico, produciendo *trayectorias* ligeramente distintas, a pesar de representar el mismo fallo. Estas situaciones se deberían tener en cuenta, variando, en la medida de lo posible, dichas condiciones para cubrir todo el rango de posibilidades dentro de un mismo comportamiento, obteniendo así un conjunto de *trayectorias* lo más heterogéneas posible para cada comportamiento.

Habitualmente, habrá más de un sensor que monitorice distintos aspectos del sistema durante el transitorio tratado, por ejemplo, valores de presión y temperatura. Se recogerán por tanto las *trayectorias* de todos los sensores que ofrezcan información del sistema, almacenando las *trayectorias etiquetadas* en *bases de datos etiquetadas* por cada variable observable.

El resultado final de este paso será una *base de datos etiquetada* por cada variable observable del sistema, que contendrá una *trayectoria etiquetada* por cada situación obtenida con la evolución de dicha variable.

El algoritmo 6.1 es el que se sigue para la obtención de la información.

Finalmente, obtendremos una *base de datos etiquetada* por cada una de los transitorios seleccionados y cada una de las variables medidas.

```
Algorithm Recogida_de_información()
  for t = 1 to total de transitorios a diagnosticar do
    for c = 1 to total de comportamientos a diagnosticar do
      for n = 1 to número de ensayos do
        if comportamiento c es reproducible then
          Reproducir el comportamiento c
        else
          Simular en el modelo el comportamiento c
        end if
        for v = 1 to total de variables observables do
          Recoger trayectoria de la variable v
          Añadir etiqueta del comportamiento c
          Almacenar la trayectoria etiquetada en la base de datos etiquetada de la
          variable v y transitorio t
        end for
      end for
    end for
  end for
```

Algoritmo **6.1:** Algoritmo para la recogida de información

6.4. Limpieza, preprocesado y tratamiento de *trayectorias*

Las *bases de datos etiquetadas*, resultantes del paso previo, contienen todas las *trayectorias* recogidas directamente de los sensores. Muchas veces, dicha información es defectuosa, bien por la excesiva presencia de ruido, bien por alteraciones puntuales en la medida, o simplemente por la excesiva información no relevante que presenta. Además, a veces, la simple medida de los sensores no es significativa para caracterizar el comportamiento del sistema. Por ejemplo, el instante en el que se alcanza el punto de operación, o la forma en que se llega a dicho punto, son más significativas que la sucesión de valores obtenidos en sí mismos.

El objetivo de este paso es doble, por una parte eliminar la gran cantidad de información no relevante presente en las *trayectorias*. Por ejemplo, en muchos sistemas la presencia de un controlador hace que todas las *trayectorias* sean muy

similares cuando se acercan al estado estacionario, independientemente de si hay fallo o no lo hay. Por otra parte, se pretende añadir a dichas *trayectorias* información no presente y que puede resultar muy útil a la hora de caracterizar su comportamiento.

En este sentido, se aplicarán técnicas de limpieza y filtrado de la información, reducción de la dimensionalidad y creación de nuevas características que proporcionen posteriormente más precisión en el proceso de aprendizaje.

6.4.1. Tratamiento de los valores ausentes

Una vez leídas las *trayectorias*, éstas deberían presentar el mismo número de medidas, que dependerán de la frecuencia de muestreo del sensor con las que se adquieren. Pueden existir ocasiones en las que dichas *trayectorias* presenten ausencias en determinados instantes. Esto es debido a que el sensor, por algún motivo, no ha recogido la medida en un determinado instante. Esto supondría un fallo del sensor. La *trayectoria* recogida no tendrá por tanto el mismo número de elementos que el resto, y por tanto no es comparable ni útil a la hora de generar el modelo.

Existen diferentes alternativas, bien tratadas en la bibliografía, sobre las diferentes posibilidades de tratamiento de los valores ausentes. Las más habituales son:

- Ignorar la ausencia, lo que supondría asumir la *trayectoria* como correcta.
- Eliminar la *trayectoria* completa.
- Eliminar el valor ausente del resto de *trayectorias*.
- Reemplazar el valor ausente por un valor aproximado que puede ser calculado mediante algún método estadístico con respecto a la propia *trayectoria* o con respecto al resto de *trayectorias*.

En el caso que nos ocupa el tratamiento seleccionado ante *trayectorias* con valores faltantes será el de la sustitución de la *trayectoria* que presente las ausencias por una nueva *trayectoria*. Esta decisión se justifica por varios motivos:

- No es posible ignorar la ausencia, puesto que la *trayectoria* no sería comparable con el resto.
- La eliminación del valor ausente del resto de *trayectorias* podría suponer una perdida de información mucho mayor que la eliminación de la propia *trayectoria*. Además, se sabe que hay un valor o varios valores ausentes, pero no la posición de dichos valores.
- Es posible reemplazar la *trayectoria* anómala haciendo una nueva captura de datos desde el sistema o el modelo del mismo sin ningún tipo de perjuicio.
- La sustitución del valor ausente, por un valor aproximado, requeriría el conocimiento específico del instante en el que se produce la pérdida, y lo único que se conoce es que faltan uno o más valores, pero no el instante de la pérdida. Por otra parte, cualquier sustitución del valor ausente supondría una aproximación no necesariamente válida. Es más, si el número de valores ausentes fuera lo suficientemente elevado supondría un problema en el sensor, y eso por si mismo es un fallo que se escapa del ámbito de aplicación de la metodología.

Así pues, cada *trayectoria* que presente valores ausentes será sustituida por otra *trayectoria* válida perteneciente al mismo fallo y transitorio.

6.4.2. Escalado y traslación del dominio

Hay ocasiones en las que puede resultar conveniente realizar una traslación del dominio de los atributos, bien de valores continuos a discretos o viceversa. En nuestro caso, las *trayectorias* están formadas por valores continuos de los atributos, que además tienen relación entre sí. Un atributo que precede a otro en la misma *trayectoria* es el valor de ese mismo atributo en el instante previo. Por tanto, la única traslación aplicable, que sería la discretización de alguno o todos los atributos no tiene sentido en el problema que nos ocupa.

Por otra parte, el escalado de las *trayectorias* provocaría que dos *trayectorias* que tengan la misma forma (relación entre sus atributos) pero con magnitudes distintas pasasen a ser prácticamente iguales. Para nuestro problema, el escalado haría que se perdiese información, puesto que dos *trayectorias* semejantes con

diferentes escalas pueden representar dos situaciones distintas (distintos fallos, o un fallo con respecto a la situación correcta). Lo que hace diferentes a dichas *trayectorias* es, precisamente, la escala a la que están, e igualarlas supondría hacer ambigua la situación. La figura 6.5 representa dos *trayectorias* iguales a distinta escala que representan dos situaciones diferentes.

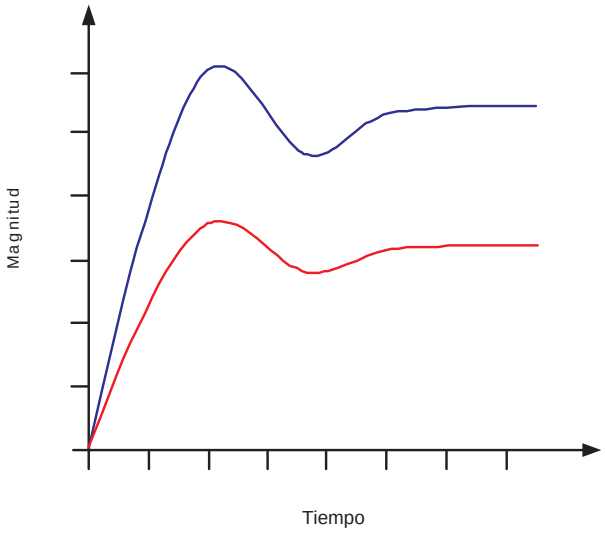

Figura 6.5: *Trayectorias* semejantes de distintos comportamientos a diferente altura

6.4.3. Filtrado del ruido

Habitualmente, las trayectorias recogidas desde los sensores suelen mostrar la presencia de ruido que alteran la medida real de la variable. Las técnicas de filtrado permiten eliminar parte de dicho ruido, aproximando la trayectoria leída a la original. Dependiendo del tipo de filtro empleado y los parámetros de ajuste del filtro, la señal original se verá alterada en mayor o menor medida. Las características que definen un filtro vienen determinadas por los siguientes conceptos:

Función de transferencia Con independencia de la realización concreta del filtro (analógico, digital o mecánico) la forma de comportarse de un filtro se describe por su función de transferencia. Ésta determina la forma en que la señal aplicada cambia en amplitud y en fase al atravesar el filtro. La función de transferencia elegida tipifica el filtro. Algunos filtros habituales son:

- Filtro de Butterworth, con una banda de paso suave y un corte agudo
- Filtro de Chevyshev, con un corte agudo pero con una banda de paso con ondulaciones
- Filtros elípticos o filtro de Cauer, que consiguen una zona de transición más abrupta que los anteriores a costa de oscilaciones en todas sus bandas
- Filtro de Bessel, que, en el caso de ser analógico, aseguran una variación de fase constante

Se puede llegar a expresar matemáticamente la función de transferencia en forma de fracción mediante las transformaciones en frecuencia adecuadas. Se dice que los valores que hacen nulo el numerador son los ceros y los que hacen nulo el denominador son polos.

$$H(f) = \frac{numerador(f)}{denominador(f)} \tag{6.4}$$

El número de polos y ceros indica el orden del filtro y su valor determina las características del filtro, como su respuesta en frecuencia y su estabilidad.

Orden El orden de un filtro describe el grado de aceptación o rechazo de frecuencias por arriba o por debajo, de la respectiva frecuencia de corte. Un filtro de primer orden, cuya frecuencia de corte sea igual a (F), presentará una atenuación de $6dB$ a la primera octava $(2F)$, $12dB$ a la segunda octava $(4F)$, $18dB$ a la tercer octava $(8F)$ y así sucesivamente. Uno de segundo orden tendría el doble de pendiente (representado en escala logarítmica). Esto se relaciona con los polos y ceros: los polos hacen que la pendiente suba con $20dB$ y los ceros que baje, de esta forma los polos y ceros pueden compensar su efecto.

Para realizar filtros analógicos de órdenes más altos se suele realizar una conexión en serie de filtros de 1 o 2 orden debido a que a mayor orden el filtro se hace más complejo. Sin embargo, en el caso de filtros digitales es habitual obtener órdenes superiores a 100.

Tipos de filtro

Atendiendo a sus componentes constitutivos, naturaleza de las señales que tratan, respuesta en frecuencia y método de diseño, los filtros se clasifican en los distintos grupos que a continuación se indica.

Según respuesta frecuencia Tendremos:

- Filtro paso bajo: Es aquel que permite el paso de frecuencias bajas, desde frecuencia 0 o continua hasta una determinada. Presentan ceros a alta frecuencia y polos a bajas frecuencia.
- Filtro paso alto: Es el que permite el paso de frecuencias desde una frecuencia de corte determinada hacia arriba, sin que exista un límite superior especificado. Presentan ceros a bajas frecuencias y polos a altas frecuencias.
- Filtro paso banda: Son aquellos que permiten el paso de componentes frecuenciales contenidos en un determinado rango de frecuencias, comprendido entre una frecuencia de corte superior y otra inferior.
- Filtro elimina banda: Es el que dificulta el paso de componentes frecuenciales contenidos en un determinado rango de frecuencias, comprendido entre una frecuencia de corte superior y otra inferior.
- Filtro multibanda: Es que presenta varios rangos de frecuencias en los cuales hay un comportamiento diferente.
- Filtro variable: Es aquel que puede cambiar sus márgenes de frecuencia.

Filtros activos y pasivos Son:

- Filtro pasivo: Es el constituido únicamente por componentes pasivos como condensadores, bobinas y resistencias.
- Filtro activo: Es aquel que puede presentar ganancia en toda o parte de la señal de salida respecto a la de entrada. En su implementación se combinan elementos activos y pasivos. Siendo frecuente el uso de amplificadores operacionales, que permite obtener resonancia y un elevado factor Q sin el empleo de bobinas.

Filtros analógicos o digitales Atendiendo a la naturaleza de las señales tratadas los filtros pueden ser:

- Filtro analógico: Diseñado para el tratamiento de señales analógicas.
- Filtro digital: Diseñado para el tratamiento de señales digitales. Entre ellos, cabe citar el Filtro Adaptado cuya función principal es maximizar la relación señal a ruido en el receptor.

El filtro a emplear dependerá del tipo de señal y ruido que se tenga, así como del compromiso entre filtrado y perdida de exactitud que se pretende alcanzar. En todo caso, es una cuestión que debe estudiarse de forma particular para cada problema.

6.4.4. Reducción de la dimensionalidad

Con frecuencia, el número de elementos que componen una *trayectoria etiquetada* es muy grande. La mayoría de las veces depende el tiempo de muestreo del sensor, que para sistemas de dinámica rápida puede dar del orden de decenas de miles de medidas por segundo. La mayoría de dicha información no es significativa desde el punto de vista de la diagnosis del sistema. Por ejemplo, los valores de las *trayectorias* que se encuentran próximos al estado estacionario, suelen ser muy similares, incluso para *trayectorias* pertenecientes a comportamientos muy diferentes desde el punto de vista del diagnóstico. Esto es debido a que la habitual presencia de controladores hace que, la mayoría de las veces, los sistemas alcancen el estado estacionario aún cuando exista un fallo en algún componente, tal como se muestra en la figura 6.6.

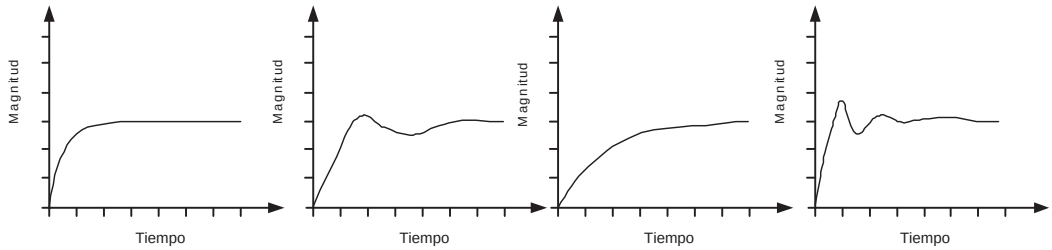

Figura 6.6: Alcance del estado estacionario bajo diferentes fallos

Este exceso de información es conocido en el proceso de $\mathcal{KDD}$ como la *maldición de la dimensionalidad*. Para problemas en los que los atributos pueden ser

de distinto tipo, o cuya relación no es temporal existen gran diversidad de técnicas que permiten reducir y seleccionar los atributos relevantes. Estas técnicas se dividen en dos grandes grupos: aquellas que realizan transformación de los atributos originales y las que únicamente realizan una selección de los más relevantes. Entre los primeros cabe destacar el *Análisis de Componentes Principales (PCA)* que consiste en transformar los atributos o variables originales $x_1, x_2, ..., x_n$ de los ejemplos en otro conjunto de atributos $f_1, f_2, ..., f_p$ donde $p \leq n$. Por su parte, las técnicas de selección de atributos fueron comentadas en el apartado 4.4.3.

Si embargo, una *trayectoria etiquetada* puede verse como un único atributo que evoluciona a través del tiempo. Por tanto, la relación entre los distintos elementos que componen la *trayectoria* es clara. Se necesita, así pues, una técnica de reducción de la dimensionalidad que elimine los elementos irrelevantes, pero conservando la esencia de la *trayectoria* en su conjunto. Si se considera una *trayectoria* como un caso específico de serie temporal, existen también numerosas técnicas que permiten reducir la dimensión de una serie temporal. Entre dichas técnicas caben destacar el uso de Transformadas de Fourier [Keogh *et al.*, 2001b], Wavelets [Chan and Fu, 1999], Mapeado Simbólido [Perng *et al.*, 2000] o Representación por Segmentos Lineales [1] [Ge and Smyth, 2001].

Las técnicas de aproximación mediante segmentos lineales han sido ampliamente usadas en el contexto de la minería de datos para diversas tareas. Intuitivamente, la aproximación por segmentos lineales consiste en aproximar una serie temporal t, de una determinada longitud n mediante k segmentos rectos. Dado que k suele ser mucho más pequeño que n, el almacenamiento, transmisión y tratamiento de dicha aproximación es más eficiente. A los algoritmos que toman como entrada una serie temporal y devuelven la representación por segmentos lineales se les suele denominar *algoritmos de segmentación.*

Los problemas de segmentación pueden ser encuadrados de varias formas:

- Dado una serie temporal t, producen la mejor representación usando únicamente k segmentos.
- Dado una serie temporal t, producen la mejor representación, tal que el máximo error de cada segmento no excede un error máximo.

[1] Picewise Linear Representation

- Dado una serie temporal t, producen la mejor representación, tal que el error acumulado de todos los segmentos no excede de un determinado límite.

Los algoritmos de segmentación pueden ser también clasificados como *batch* u *on-line*. Esta distinción es importante, puesto que los procesos dinámicos suelen requerir procesos *on-line*.

Antes de proponer un método de reducción del tamaño de dichas *trayectorias* necesitaremos algunas definiciones:

Definición 6.4. **Trayectoria canónica**. Una *trayectoria canónica* es una *trayectoria etiquetada*, de la cual se han eliminado los elementos no significativos. Puede ser denotada por:

$$c_i = \prod_{\Gamma'} j_i \tag{6.5}$$

donde $\Gamma' \subset \Gamma$. Recordemos que Γ es un conjunto finito de k instantes de tiempo que tienen los siguientes valores:

$$\Gamma = \{t_0, t_1 = t_0 + \Delta t, t_2 = t_0 + 2\Delta t, \ldots, t_{k-1} = t_0 + (k-1)\Delta t\} \tag{6.6}$$

donde Δt viene dado por la frecuencia de muestreo de la variable observable.

Definición 6.5. **Base de datos canónica** Una *base de datos canónica* es una *base de datos etiquetada* en la cual, cada *trayectoria etiquetada* ha sido sustituida por su correspondiente *trayectoria canónica*. Denotaremos por C a la *base de datos canónica*.

El número de elementos que componen una *trayectoria canónica* dependerá de la forma de dicha *trayectoria* y por tanto de lo significativos que sean los elementos que la componen. Así pues, una *base de datos canónica* C es un conjunto de *trayectorias canónicas* donde cada *trayectoria canónica* está compuesta por un número diferente de elementos.

Definición 6.6. **Base de datos normalizada**. Una *base de datos normalizada* N es una *base de datos canónica* donde cada *trayectoria* está formada por el

mismo número de elementos. Todos los elementos con el mismo cardinal en cada *trayectoria* de la *base de datos normalizada* se corresponden con el mismo instante de tiempo.

Definición 6.7. **Trayectoria normalizada**. Una *trayectoria normalizada* es una *trayectoria* de una *base de datos normalizada*.

Tanto las *bases de datos canónicas*, como las *bases de datos normalizadas*, son casos particulares de la definición más general de *base de datos transformada*.

Para conseguir transformar cada una de las *trayectorias etiquetadas* en *trayectorias canónicas*, se ha seleccionado un algoritmo de segmentación. El objetivo de dicho algoritmo es caracterizar cada una de las *trayectorias etiquetadas* mediante una sucesión de segmentos rectos. Esta sucesión de segmentos rectos aproxima la *trayectoria* con muchos menos puntos que la *trayectoria* original. En [Keogh *et al.*, 2001a] se puede encontrar una comparativa entre diferentes algoritmos de segmentación. El algoritmo seleccionado ha sido el algoritmo de la ventana deslizante. La razón para seleccionar dicho algoritmo es que puede trabajar sin necesidad de conocer la *trayectoria* completa, sino únicamente los puntos tratados. Esto lo hace ideal para el caso que nos ocupa, puesto que hace que el algoritmo pueda trabajar *on-line*, mientras se están recogiendo los datos de los sensores en el sistema monitorizado. En la fase *off-line* de la metodología esto no es necesario, puesto que se puede esperar a tener la *trayectoria* completa para realizar el tratamiento, pero en la fase *on-line*, donde se evalúa la nueva observación, la segmentación deberá realizarse mientras ésta está siendo recogida, con objeto de tener una diagnosis en tiempo real. Este es el motivo de la elección de dicho algoritmo.

El algoritmo 6.2 muestra la forma de segmentación mediante ventana deslizante.

La figura 6.7 muestra como el algoritmo aproxima una *trayectoria* mediante una sucesión de segmentos rectos.

En este algoritmo, el segmento está creciendo mientras no se traspase un determinado umbral de error.

$$\sum_{i=1}^{n} |x_i - s_i| < error \tag{6.7}$$

Algorithm T_Seg = Ventana_deslizante(T, $error\_max$)
  $ancla = 1$
  while NOT finalice la trayectoria T **do**
    $i = 2$
    while calcula_error($T[ancla : ancla + i]$) $< error\_max$ **do**
      $i = i + 1$
    end while
    T_Seg = **concatena**(T_Seg , crea_segmento($T[ancla : ancla + (i - 1)]$)
    $ancla = ancla + i$
  end while

Algoritmo **6.2:** Algoritmo de segmentación de ventana deslizante

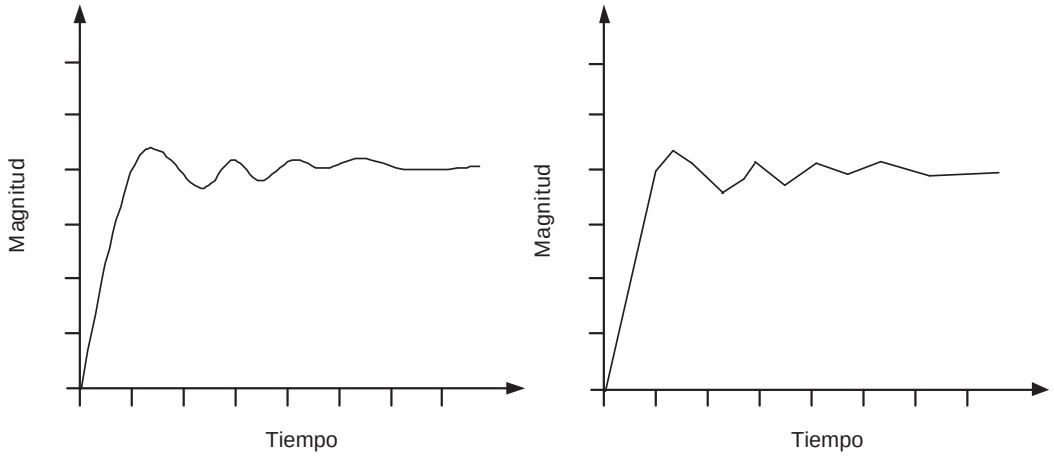

Figura 6.7: Aproximación de una *trayectoria* mediante segmentos

donde x_i es el valor de la *trayectoria etiquetada*, s_i es el valor del segmento generado y n representa el número de elementos del segmento.

Cuando el umbral del error definido es superado, se comienza a formar un nuevo segmento, partiendo del instante anterior al que traspasó el umbral de error. El error se calcula usando la ecuación 6.8.

$$Error = \lambda \sqrt{\sum_{i=1}^{n} \frac{(x_i - s_i)^2}{n}} + (1 - \lambda) \max |x_i - s_i|, \quad \lambda \in [0, 1] \tag{6.8}$$

El primer sumando representa la desviación acumulada entre la sección de la *trayectoria etiquetada*, que se está tratando, y el segmento en curso. El segundo

sumando controla que la desviación en un punto concreto no supere el error. Cada uno de los términos puede ser ponderado mediante el factor λ, según la aproximación que se pretenda conseguir entre el segmento y la *trayectoria etiquetada*.

La sucesión de segmentos que aproximan la *trayectoria etiquetada* formarán la *trayectoria canónica*.

El algoritmo de segmentación nos proporciona dos ventajas:

1. Cada *trayectoria etiquetada* es representada usando mucha menos información. Esto facilita el almacenamiento de las *trayectorias canónicas* y su manipulación.

2. Es posible usar eficientemente el valor de la derivada de cada uno de los segmentos que forman la *trayectoria canónica*. Al estar formadas dichas *trayectorias* por segmentos rectos, la pendiente de cada segmento será la derivada en cada uno de los puntos de dicho segmento. Esta propiedad será muy útil en la sección 6.4.5.

Normalización de atributos

En el proceso de segmentación se han conseguido las *bases de datos canónicas*, pero el problema que se plantea es que, dichas bases de datos deberán servir como base para el proceso de aprendizaje que nos permita extraer el modelo proposicional que caracteriza el comportamiento del sistema. Para que el proceso de aprendizaje pueda ser llevado a cabo, necesitaremos que cada una de las *trayectorias canónicas* esté compuesta por el mismo número de elementos, y que todos los elementos de dichas *trayectorias canónicas* se correspondan con instantes de tiempos similares. Esto no ocurre con las *trayectorias* de las *bases de datos canónicas*. Para solucionar esto, usaremos un algoritmo de normalización. Dicho algoritmo se muestra en 6.3.

El objetivo final del algoritmo de normalización es conseguir que todas las *trayectorias canónicas* de una *base de datos canónica* estén formadas por el mismo número de elementos, que se correspondan con los mismos instantes de tiempo. Es decir, transformará una *base de datos canónica* en una *base de datos normalizada*.

El algoritmo realiza dos iteraciones sobre la *base de datos canónica*. En la primera iteración, se recogen todos los instantes de tiempo donde alguna *trayectoria*

Algorithm $BD\_Normalizada$ = Normaliza(BD_Base)
  $instantes = \{\}$
  $trayectoria$ = leer desde BD_Base
  while $\exists$ trayectorias en BD_Base **do**
    for $s = 1$ **to** total de segmentos de $trayectoria$ **do**
      if instante_finalización_segmento(s) $\notin instantes$ **then**
        $instantes = instantes \bigcup$ instante_finalización_segmento(s)
      end if
    end for
    $trayectoria$ = leer desde BD_Base
  end while
  ir al comienzo de la BD_Base
  $trayectoria$ = leer desde BD_Base
  while $\exists$ trayectorias en BD_Base **do**
    for $t = 1$ **to** total de entradas de $instantes$ **do**
      if $\nexists$ valor($trayectoria$) para $instantes(t)$ **then**
        genera nuevo valor para $trayectoria$ en $instantes(t)$
        inserta el nuevo valor en $trayectoria$
      end if
    end for
    almacenar $trayectoria$ en $BD\_Normalizada$
    $trayectoria$ = leer desde BD_Base
  end while

Algoritmo **6.3:** Algoritmo de normalización

canónica tenga el comienzo o final de un segmento. En la segunda iteración, se generan nuevos segmentos para todos los instantes de tiempo recogidos en la iteración anterior y para todas las *trayectorias* que no tuvieran un comienzo o final de segmento en dichos instantes de tiempo. Cada *trayectoria normalizada* se va almacenando para formar la *base de datos normalizada*. La generación de nuevos segmentos a partir de los ya existentes es una tarea fácil y rápida. Sólo es necesario aplicar la ecuación de la recta $y = mx + n$, se calculan m y n para el segmento actual y a partir de ellos se calcula el nuevo valor de y para el nuevo instante de tiempo x.

El coste que se debe pagar por el proceso de normalización, es que la dimensión de la *base de datos normalizada* será siempre superior al de la *base de datos canónica*. Esto es inevitable, al tener que generar nuevos segmentos en puntos intermedios de segmentos ya existentes. No obstante la nueva *base de datos normalizada* consigue reducir la dimensión con respecto a la *base de datos etiquetada* original. Esto se hace posible debido a que al acercarse al estado estacionario, todas las *trayectorias* son muy similares, y por tanto el proceso consigue una reducción muy importante en esos casos.

6.4.5. Cálculo de nuevos atributos y valores relevantes

Muchas veces, la información recogida directamente de los sensores es insuficiente para diagnosticar el comportamiento del sistema. Existen una serie de valores relacionados con la *trayectoria* recogida por el sensor que pueden ser más significativos desde el punto de vista de la diagnosis. A continuación expondremos algunos de ellos.

Especificaciones de la respuesta temporal

La respuesta en el tiempo de un sistema consiste de 2 partes:

- Respuesta transitoria: parte de la respuesta total que tiende a cero a medida que el tiempo tiende a infinito.
- Respuesta estacionaria o permanente: parte de la respuesta total que no tiende a cero a medida que el tiempo tiende a infinito.

Se define el orden de un sistema cuya función de transferencia es

$$G(s) = \frac{n(s)}{d(s)} \tag{6.9}$$

como el grado del polinomio denominador $d(s)$.

Si el sistema que vamos a diagnosticar puede ser tratado como un sistema de segundo orden, existen algunas especificaciones que pueden ser muy útiles a la hora de caracterizar la *trayectoria*. Un sistema de segundo orden tiene como función de transferencia a la siguiente ecuación:

$$\frac{Y(s)}{U(s)} = \frac{\omega_n^2}{s^2 + 2\xi\omega_n s + \omega_n^2} \tag{6.10}$$

donde: ω_n es la frecuencia natural no amortiguada y ξ el coeficiente de amortiguamiento.

La respuesta del sistema depende de las raíces del denominador (polos del sistema). Para un sistema de segundo orden los polos se expresan como:

$$-\delta\omega_n \pm \omega_n\sqrt{\delta^2 - 1} \tag{6.11}$$

Dependiendo del valor de δ podemos tener los siguientes 3 casos:

1. $0 < \delta < 1$, polos complejos conjugados en la parte izquierda del plano complejo. En este caso se dice que el sistema es subamortiguado.

2. $\delta = 1$, polo real repetido. Se dice que el sistema tiene amortiguamiento crítico.

3. $\delta > 1$, polos reales distintos. El sistema se dice sobreamortiguado.

Tal como se observa en la figura 6.8 cuando $\delta = 0$ las oscilaciones continuarán indefinidamente. Para valores mayores de δ se obtiene un decaimiento más rápido de las oscilaciones, pero con un ascenso más lento de la respuesta. En el caso en el que $\delta = 1$, el sistema se torna críticamente amortiguado a tal punto que desaparecen las oscilaciones. Valores mayores de δ producen un sobreamortiguamiento del sistema.

En la teoría de control, la caracterización de la respuesta temporal de un sistema se suele hacer mediante lo que se conoce como especificaciones del sistema. Las especificaciones más empleadas son:

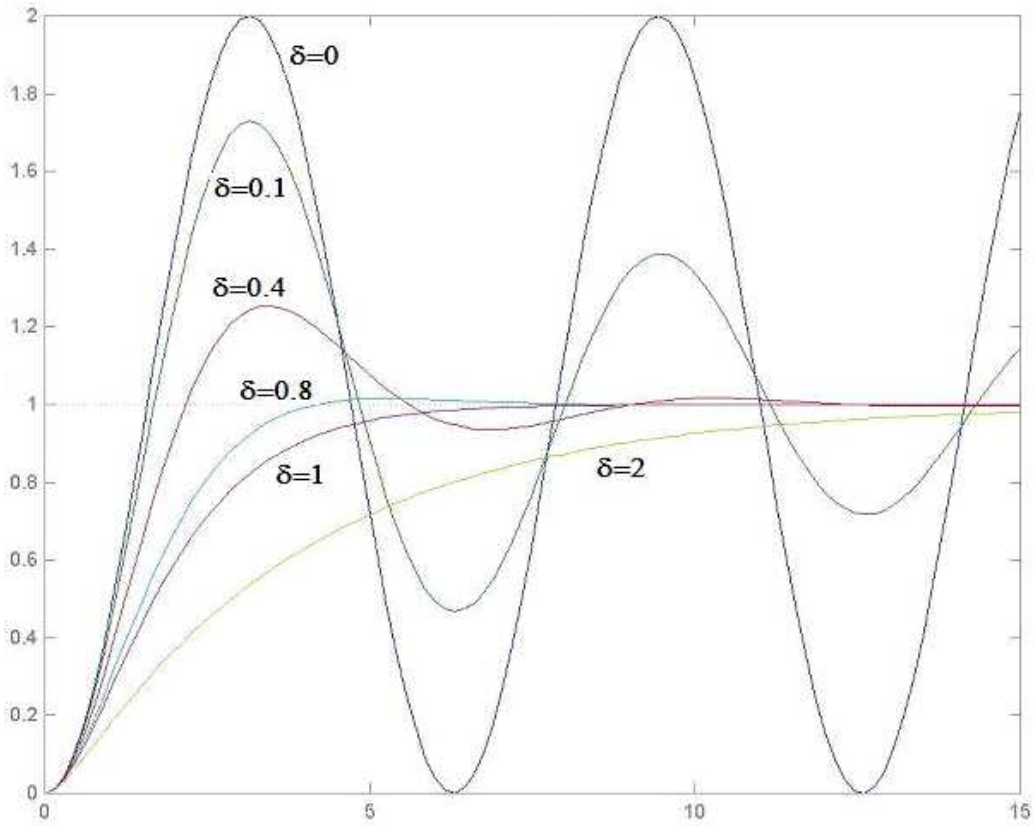

Figura 6.8: Comportamientos de un sistema de 2^{o} orden

- Tiempo de retraso (delay time, t_d). Es el tiempo necesario para que la respuesta alcance el 50 % del valor final.

- Tiempo de subida (rise time, t_r). tiempo requerido para que la respuesta crezca del 10 % al 90 % (sobreamortiguado), del 5 % al 95 %, o del 0 al 100 % (subamortiguado) de su valor final.

$$t_r = \frac{1}{\omega_d} \tan^{-1}\left[-\frac{\sqrt{1-\delta^2}}{\delta}\right] \tag{6.12}$$

- Tiempo de pico (t_p): Es el tiempo que pasa hasta alcanzarse el primer pico de sobreimpulso.

$$t_p = \frac{\pi}{\omega_d} \tag{6.13}$$

- Sobreimpulso máximo (M_p): Máximo sobreimpulso es el valor pico máximo de la curva de respuesta medido desde la unidad.

$$M_p = e^{-(\delta/\sqrt{1-\delta^2})\pi} \tag{6.14}$$

Si el valor final de la respuesta es diferente de 1, se utiliza el máximo sobreimpulso porcentual, que está dado por:

$$M_p = e^{-(\delta/\sqrt{1-\delta^2})\pi} \times 100\,\% \tag{6.15}$$

- Tiempo de establecimiento (t_s): tiempo requerido por la curva de respuesta para alcanzar y mantenerse dentro de determinado rango alrededor del valor final, especificado en porcentaje absoluto del valor final (se usa generalmente el 5 % o el 2 %).

$$\text{Criterio del } 2\,\% \qquad t_s = \frac{4}{\delta\omega_n} \tag{6.16}$$

$$\text{Criterio del } 5\,\% \qquad t_s = \frac{3}{\delta\omega_n} \tag{6.17}$$

Tendencias

Consideraremos la tendencia de una *trayectoria* como la forma en que dicha *trayectoria* evoluciona en el tiempo, independientemente de los valores absolutos con los que evolucione. Esto es, dos *trayectorias* que tiendan a subir en los primeros instantes de tiempo tendrán la misma tendencia en esos instantes, independientemente de los valores de la medida del sensor.

Para poder usar dicha información en el proceso de diagnosis, necesitaremos añadir dicha información a la *trayectoria*, con objeto de que pueda ser usada posteriormente. Con este fin, se realiza la siguientes definición:

Definición 6.8. **Comportamiento perfecto**. La *trayectoria* que describe el sistema cuando todos sus componente funcionan de forma ideal, sin desviarse de sus valores nominales.

En base a dicha definición podemos calcular las siguientes magnitudes:

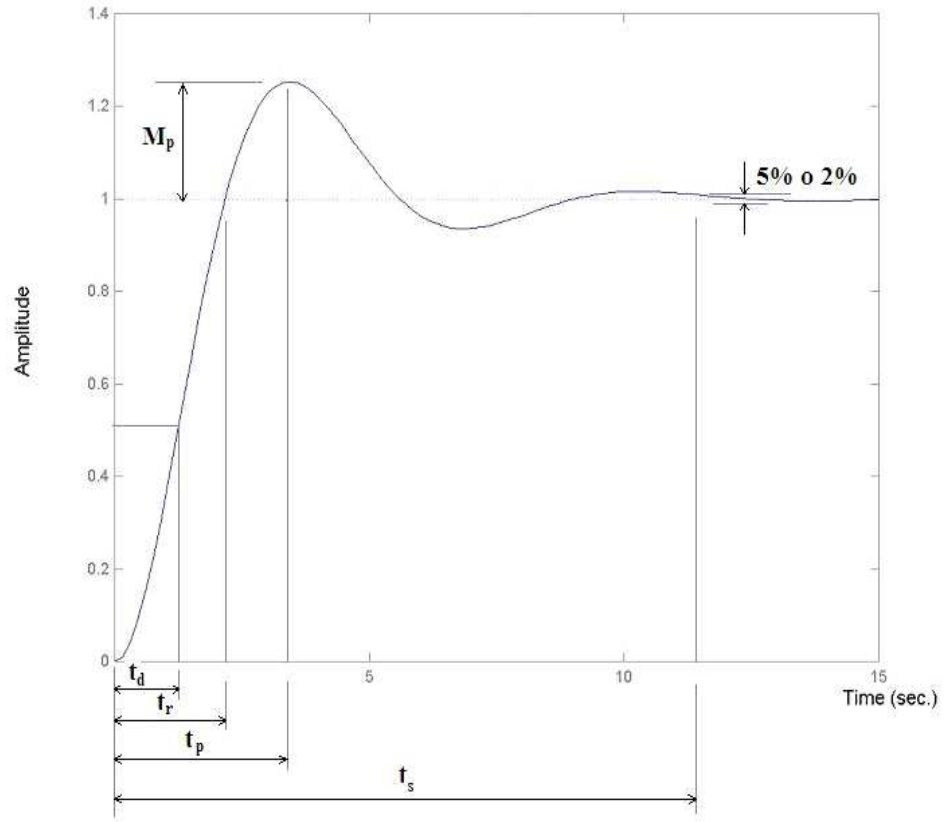

Figura 6.9: Especificaciones de un sistema de 2^{o} orden

- Inclinación relativa (IR). Se calcula para cada uno de los elementos de la *trayectoria*. Determina el grado de inclinación con que la *trayectoria* llega a ese punto, independientemente de su posición absoluta. Con esto conocemos la pendiente que forma la recta que une dicho elemento con el elemento inmediatamente anterior. Será una medida de lo rápido que aumenta o disminuye la medida del sensor en dicho punto. Representa la derivada de la *trayectoria* en el punto en que se trata. Se calcula como:

$$IR(i) = \frac{V_m[i] - V_m[i-1]}{\Delta t} \tag{6.18}$$

donde $V_m[i]$ representa el valor de la *trayectoria* en el instante i.

Cuando se realiza el cálculo de esta magnitud tras el proceso de segmentación de la *trayectoria* (sección 6.4.4, en realidad estamos obteniendo la pendiente del segmento a la que pertenece dicho elemento de la *trayectoria*.

- Distancia al comportamiento perfecto (DP). Se calcula para cada uno de los elementos de la *trayectoria*. Determina lo lejos o cerca que nos encontramos del comportamiento perfecto. Con esta magnitud se pretende obtener una medida de proximidad o alejamiento, del comportamiento actual, al comportamiento que tendría el sistema si todos los valores de las constantes de los componentes tomaran el valor central del rango en que se considera que el comportamiento del componente es correcto. Se calcula como:

$$DP(i) = V_m[i] - V_{mpf}[i] \tag{6.19}$$

donde $V_{mpf}[i]$ es el valor de V_m en el punto i para la *trayectoria* perfecta.

- Inclinación relativa a la *trayectoria* perfecta(IRP). Se calcula para cada uno de los elementos de la *trayectoria*. Determina la inclinación de acercamiento o alejamiento a la *trayectoria* perfecta. Esta medida proporciona, para cada uno de los puntos de la muestra seleccionada, la pendiente que tendrá, en cada punto, la *trayectoria* formada por los puntos que conforman la *trayectoria* de la distancia al comportamiento perfecto. Se define como:

$$IRP(i) = \frac{DP[i] - DP[i-1]}{\Delta t} \tag{6.20}$$

- Cambios de tendencia. Se calcula para toda la *trayectoria*. Indica el número de veces que ha cambiado de signo la *inclinación relativa* de la *trayectoria*. Indica el grado de inestabilidad de la *trayectoria*. Una *trayectoria* con un valor alto en los cambios de tendencia, estará realizando una aproximación a la referencia de una forma mucho más inestable que otra con menor número de cambios de tendencia.

6.5. Generación del modelo proposicional

El resultado de toda la etapa de tratamiento de las *trayectorias* es un conjunto de *bases de datos normalizadas*, que contienen *trayectorias normalizadas* de todos los comportamientos. Existe una *base de datos normalizada* por cada transitorio tratado y cada variable recogida.

El paso previo a la generación del modelo proposicional es realizar la unión de todas las *bases de datos normalizadas* correspondientes al mismo transitorio en una única *base de datos del transitorio*.

Definición 6.9. **Base de datos del transitorio**. Denotamos por N_{tv} la *base de datos normalizada* correspondientes al transitorio t y la variable v. Sea b un determinado comportamiento a diagnosticar, denotamos por n_{tvbi} a la *trayectoria normalizada* correspondiente al transitorio t, variable s y comportamiento b obtenida en la captura i. Para el transitorio t la *base de datos del transitorio* T contiene la concatenación de todas las *trayectorias* del mismo comportamiento b con y la misma captura i de cada una de las *bases de datos normalizadas* de dicho transitorio.

La *Base de datos del transitorio* contiene la concatenación de cada una de las *trayectorias normalizadas* correspondientes al mismo transitorio, pero a variables distintas. El último atributo de cada entrada resultante será la etiqueta representativa del comportamiento al que caracterizan. Una *base de datos del transitorio* es un tipo de *base de datos transformada*.

La *base de datos del transitorio* cumple todas las condiciones para poder realizar aprendizaje automático a partir de la información que contiene. Puesto que se conoce la clase de cada una de las instancias de dicha base de datos (la etiqueta correspondiente al comportamiento), el **aprendizaje supervisado** es la técnica apropiada para extraer información a partir de dichos datos.

La elección de la herramienta de clasificación usada viene determinada por diversos factores intrínsecos al problema que nos ocupa y a los objetivos que se pretenden conseguir.

- El modelo obtenido debería ser inteligible por los humanos, para poder así detectar cual es la causa que provoca el fallo y actuar en consecuencia.

- La representación del modelo debería poderse almacenar en poco espacio, lo que permitirá usar dispositivos específicos de coste limitado.
- La evaluación del modelo debería hacerse de forma muy rápida, lo que requeriría un coste computacional pequeño a la hora de evaluar una nueva observación. Por otra parte, un tiempo de evaluación elevado provocaría que los sistemas de dinámica muy rápida no pudiesen ser diagnosticados *on-line*.

Todos estos condicionantes hacen que la herramienta de clasificación seleccionada sean los **árboles de decisión**. El motivo por el cual se han seleccionado los árboles de decisión es que el diagnóstico puede ser alcanzado en muy poco tiempo, debido al bajo coste computacional necesario para evaluar estos árboles y la poca necesidad de almacenamiento que tienen. Esto los hace ideales para diagnosticar sistemas de dinámica muy rápida, donde la frecuencia de muestreo es muy alta y los fallos evolucionan muy rápidamente. Esta simplicidad permite, además, que el mecanismo de diagnosis pueda ser implementado y almacenado con unos requisitos técnicos y económicos muy bajos. Por otra parte, los árboles de decisión trabajan muy bien con grandes cantidades de información, tanto si se trata de un gran número de atributos (dimensionalidad), como si se trata de una gran cantidad de ejemplos (cardinalidad). Además, son fáciles de usar, soportan bien la presencia de ruido, son tolerantes a la presencia de atributos no significativos y poseen una representación simbólica. Por contra, aprenden peor y son menos precisos en el aprendizaje que otros métodos, aunque esto puede ser paliado mediante el uso de otras técnicas como el *boosting* o el *bagging*. Diversos trabajos han demostrado la validez de los árboles de decisión aplicados a la diagnosis [Abad *et al.*, 2002; 2005].

Los árboles de decisión son una forma de representar funciones discretas $f(\omega)$ de múltiples variables ω que pueden ser instanciados con diferentes valores reales. El valor de la función será la hoja del árbol, que para el caso que nos ocupa es un comportamiento del sistema.

Este será el último paso de la fase *off-line*, que generará como resultado el árbol de decisión que servirá como módulo de diagnosis.

La tabla 6.1 muestra un ejemplo de árbol de decisión. Puede observarse como las condiciones de los nodos se corresponden con valores de las *trayectorias* en distintos instantes de tiempo, mientras que las hojas del árbol están compuesta

por la etiqueta del conjunto de fallos B, que caracteriza el comportamiento de la *trayectoria* para los valores de los nodos superiores. Así por ejemplo, $val(4)$ representa el valor de la *trayectoria* en el instante 4, que será el primer valor seleccionado por el clasificador para asignar la clase.

```
if Val(4) <= 3.63
   if Val(7) <= 0.01
      FAULT = D1
   else
       if Val(7) <= 0.2589
          FAULT = D2
       else
          FAULT = IG
       end
   end
else
   if Val(2) <=4.8
       if Val(9) <= 35.83
          FAULT = F1
       else
          FAULT = RD
       end
    else
       if Val(119) <= 7.89
          FAULT = BF
       else
          if Val(5) <= 0.704
             NO FAULT
          else
             FAULT = BA
          end
       end
    end
end
```

Tabla 6.1: Ejemplo de árbol de decisión

Antes de seguir adelante con la generación del árbol, necesitamos dar algunas definiciones.

Definición 6.10. **Modelo proposicional**. Es un árbol de decisión, o conjunto de árboles, obtenidos mediante clasificación de la *base de datos del transitorio*, de tal forma que dichas estructuras representen una función $f : T \rightarrow B$.

Definición 6.11. **Modelo observacional** (OM). Un modelo observacional es una nueva *trayectoria*, obtenida de los sensores del sistema, y tratada en el mismo sentido en que lo fueron las *trayectorias* a partir de las cuales se obtuvo el modelo proposicional.

Definición 6.12. **Diagnosis**. La diagnosis, es la evaluación de la función f del modelo proposicional, representado como un árbol de decisión o conjunto de árboles. Esta función, devuelve un comportamiento. Por esta razón:

$$Diagnosis \equiv B = f(OM) \tag{6.21}$$

donde OM es el modelo observacional.

Así pues, el objetivo final de la fase *off-line* de la metodología es conseguir el modelo proposicional del sistema. Para ello, se aplicará una herramienta de clasificación a la *base de datos del transitorio*, con objeto de lograr el árbol o conjunto de árboles representativos del modelo proposicional.

La clase de cada una de las *trayectorias* será su etiqueta. Cada elemento de la *trayectoria* será considerado un atributo del ejemplar a clasificar. Se compararán, por tanto, los mismos atributos de todos los ejemplares para conseguir la regla que sea capaz de clasificar las diferentes clases.

El árbol de decisión obtenido, representa aquellos instantes de tiempo donde las *trayectorias* de diferentes clases, es decir, correspondientes a comportamientos distintos, puedan ser clasificadas en base a un determinado valor. El valor de cada una de los nodos del árbol será una condición compuesta por un valor de la lectura y un instante de tiempo. Las hojas del árbol representan la clase, es decir, la etiqueta de la *trayectoria* que identifica el comportamiento del sistema.

Como ejemplo tomemos el siguiente nodo que evalúa dos hojas del árbol:

```
if v(14) > 12.21 then
    Clase1
else
    Clase2
```

donde la condición $v(14)$ representa el instante de tiempo 14 y el valor 12,21 el valor que es capaz de diferenciar en dicho instante a los ejemplares correspondientes a las clases *Clase*1 y *Clase*2. La figura 6.10 muestra como dicha condición, con ese valor en dicho instante de tiempo es capaz de clasificar los dos conjuntos de *trayectorias* pertenecientes a las distintas clases.

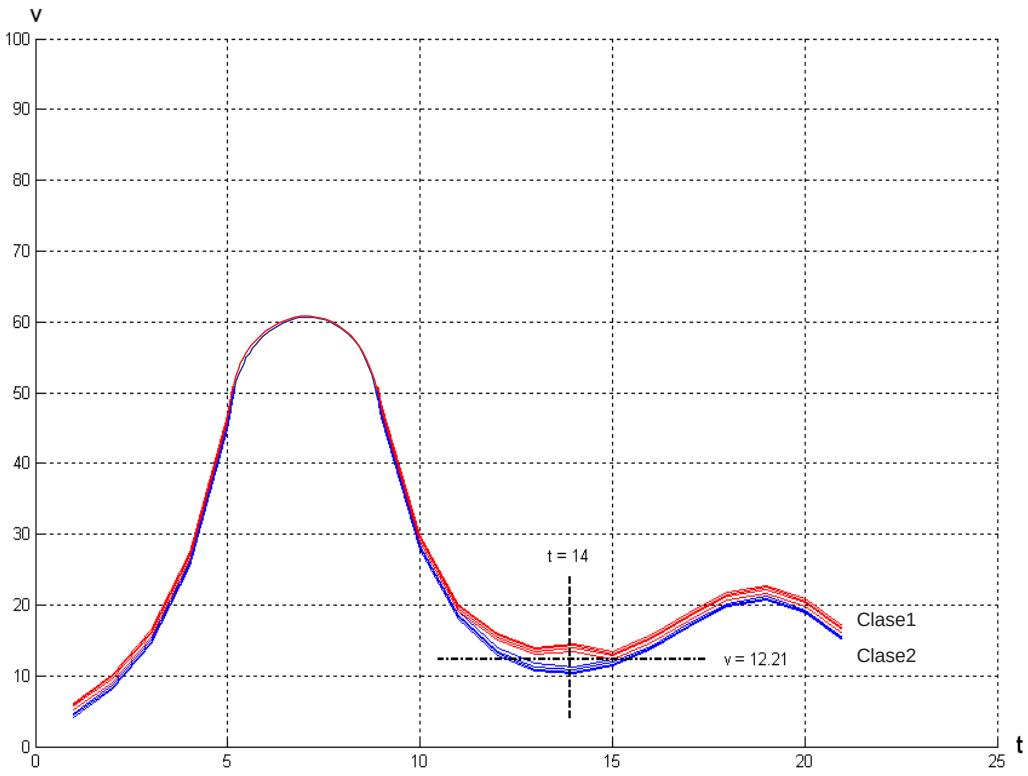

Figura 6.10: Representación del nodo que clasifica las clases *Clase1* y *Clase2*

Una de las principales ventajas que se obtienen al utilizar los árboles de decisión es que, al ser éstos inteligibles, se pueden evaluar la nueva observación conforme va alcanzando los instantes para los cuales se han generado hojas del árbol. Esto permite que la diagnosis se puede realizar *on-line* mientras el sistema evoluciona a través del transitorio, y no es necesario esperar a la finalización del mismo para alcanzar el diagnóstico. Los tratamientos aplicados a las *trayectorias*, y que también hay que aplicar a la nueva observación, no suponen un problema, puesto que éste fue el objetivo que se persiguió al utilizar el algoritmo de segmentación de la ventana deslizante. La segmentación se puede hacer mientras la *trayectoria* está siendo obtenida, y no es necesario esperar a que finalice su captura.

6.5.1. Mejorando la precisión: Boosting y Bagging

Los árboles de decisión son algoritmos de aprendizaje débiles, lo que provoca que los árboles obtenidos sean muy sensibles a los datos de entrenamiento usados.

Para mejorar la precisión y robustez de los resultados usaremos multiclasificadores. En la sección 5.5 se expusieron las tres técnicas más usadas: *Bagging*, *Boosting* y *Randomization*, siendo los dos primeros los más extendidos.

Una importante ventaja de los métodos multiclasificadores es que la combinación de varios modelos semánticamente diferentes subsana el problema del sobreajuste de los datos de entrenamiento.

La idea básica consiste en que, dado un conjunto de clasificadores, se pueden conseguir mejores resultados usando una combinación de todos ellos, en vez de usarlos individualmente, incluso si es el mejor.

La diferencia entre *bagging* y *boosting*, es que en *bagging* no se tienen en cuenta los resultados de cada uno de los subconjuntos clasificados a lo hora de generar el siguiente, mientras que *boosting* asigna más importancia a los casos mal clasificados en cada iteración con objeto de mejorar los resultados de la siguiente.

En teoría, el *bagging* y el *boosting* pueden ser usados para reducir significativamente el error de cualquier algoritmo de clasificación que genere, de forma consistente, clasificadores que sean sólo un poco mejores que la estimación al azar.

En cualquier caso, el resultado del proceso de clasificación generará tantos árboles de decisión como clasificaciones realizadas. Por tanto, el conjunto de dichos árboles conformarán el modelo proposicional (fig 6.11).

En el caso de que se haya usado *bagging*, el peso de todos los árboles generados será homogéneo, mientras que para el *boosting* el peso de cada uno de los árboles es inversamente proporcional al error estimado.

Evaluación

Puesto que el proceso de Multiclasifiación se ha usado para elaborar un modelo proposicional con tantos árboles de decisión como clasificaciones efectuadas, la evaluación del modelo observacional deberá realizarse evaluando cada uno de los árboles del modelo proposicional (fig 6.12). Cada una de dichas evaluaciones ofrecerá una diagnosis, pero el diagnóstico deberá ser único. De esta forma, se realizará una votación entre todos los diagnósticos parciales para obtener el diag-

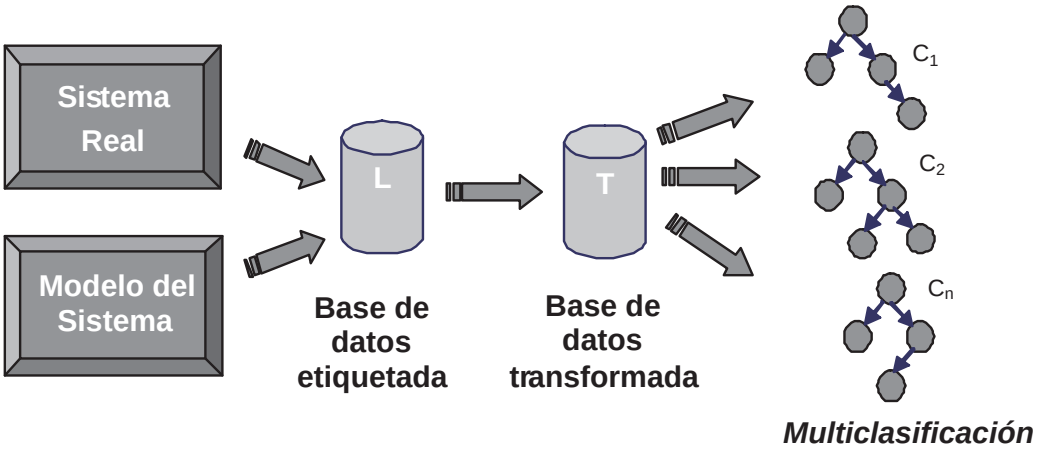

Figura 6.11: Multiclasificación

nóstico final. Para el caso en el que se haya usado *bagging* para generar los árboles, el diagnóstico se corresponderá con la clase más votada, mientras que si se ha usado *boosting*, cada clasificación individual es ponderada con el peso correspondiente al árbol que la ofrece. Se usará un número impar de árboles con objeto de deshacer los posibles empates en la votación.

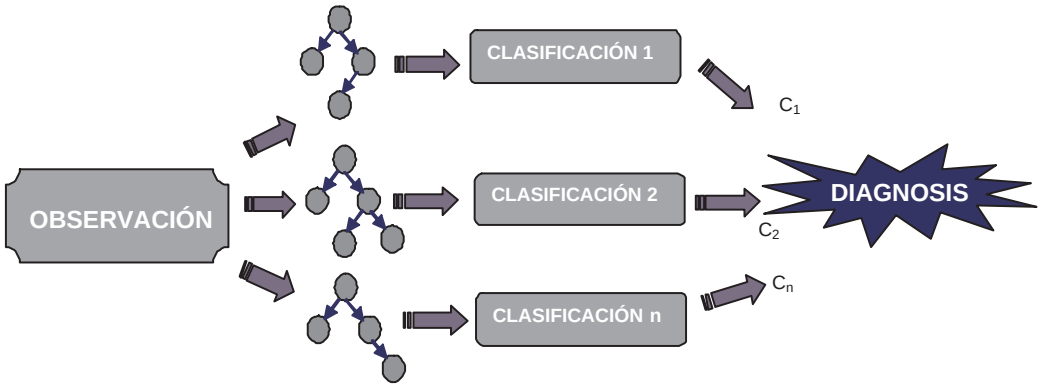

Figura 6.12: Evaluación de la Multiclasificación

El resultado de la votación será el diagnóstico obtenido.

6.6. Validación de resultados

Una vez generados los árboles de decisión con los que realizar la diagnosis, y antes de pasar a la implantación en el sistema se deberá valorar la validez de los resultados obtenidos. Este paso se realizará en dos etapas:

- La primera validación que se realiza sobre los árboles obtenidos consiste en hacer una validación cruzada sobre los datos de entrenamiento. Esto nos proporcionará una primera idea de la posibilidad de realizar una clasificación lo suficientemente precisa con los datos disponibles.

- Posteriormente, se usará el conjunto completo de datos de entrenamiento para generar los árboles de decisión y se obtendrán nuevos datos del sistema con los que evaluar los árboles obtenidos. Estos nuevos datos no estaban presentes en el conjunto de entrenamiento, y por tanto representan un test bastante fiable de la validez de los árboles obtenidos.

Si los resultados de la validación son óptimos se procederá a la fase de implantación, durante la cual se realiza una validación continua, sometiendo al sistema a diversos ensayos para comprobar la validez del módulo de diagnosis.

Si la validación no resulta aceptable en cualquiera de sus fases, se deberán generar nuevos árboles de decisión más precisos. Para ello tenemos varias alternativas:

- Comenzar la metodología de nuevo, añadiendo más *trayectorias* de cada comportamiento. Esto paliaría el problema si la falta de precisión viene dada por la insuficiencia de la información recogida.

- Mantener las *trayectorias* originales, pero ajustar los parámetros, tanto en la fase de tratamiento de *trayectorias* como en la generación de los árboles. El ajuste de parámetros en la fase de tratamiento vendría dada básicamente por realizar una segmentación más fina, que se ajuste mejor a la *trayectoria* original, a costa de aumentar la dimensionalidad. El ajuste en la generación de los árboles consistiría en aumentar él número de iteraciones usadas en la multiclasificación o modificar la técnica de clasificación usada.

- Hay ocasiones en las que la clasificación no puede mejorarse más debido a que el patrón que siguen los datos no permite una mejor clasificación. Es este caso, las clases no son diferenciables entre sí, y el problema no es del clasificador sino intrínseca de los propios datos. En estos casos deberemos emplear las técnicas presentadas en la sección siguiente.

6.7. Alternativas al problema de la diagnosticabilidad

Un factor clave para la diagnosis práctica en la industria es el hecho de instalar el mínimo número de sensores, pero obteniendo un alto grado de detección de fallos y diagnosis. Para contener los costes los sistemas industriales son configurados típicamente con el mínimo número de controles necesarios para el control y la protección. La experiencia muestra que este conjunto estándar de sensores tiene muchas limitaciones a la hora de diagnosticar correctamente los fallos.

Estudiar la diagnosticabilidad de un sistema se corresponde con el estudio de si dicho sistema puede ser diagnosticado con los sensores disponibles actuales y el número de lecturas de dichos sensores. De esta forma se decide el número de fallos que es posible diagnosticar y distinguir en el sistema con el número de sensores disponibles y la colocación de dichos sensores. Esta es una propiedad que será muy importante en diferentes tareas, por ejemplo durante el proceso de diseño de un sistema puede ser utilizado para decidir el número y tipo de sensores, o en el proceso de desarrollo de software para la diagnosis puede ser usado para evaluar distintas alternativas de modelado.

En [Scarl, 1994] podemos encontrar un análisis de cómo encontrar la ubicación optima de los sensores para conseguir la diagnosticabilidad de un sistema.

En [Console *et al.*, 2000] podemos encontrar un estudio de la diagnosis y diagnosticabilidad de un sistema a través del álgebra de procesos (process algebra), más concretamente PEPA [2] , en el ámbito de la *diagnosis basada en modelos*. En este artículo podemos encontrar la siguiente definición de diagnosticabilidad: *Se dice que un sistema es diagnosticable con un conjunto de sensores si y solo si:*

[2]Performance Evaluation Process Algebra

I *Para cualquier combinación relevante de lectura de los sensores hay sólo un candidato mínimo para la diagnosis.*

II *Todos los fallos del sistema pueden ser explicados con un candidato de diagnosis para alguna lectura de los sensores.*

En [Travé-Massuyes *et al.*, 2001] se aplica la diagnosticabilidad y colocación de sensores basada en modelos a un subsistema de una turbina de gas. Aquí podemos encontrar una definición de *diagnosticabilidad parcial*, que extiende a la proporcionada en [Console *et al.*, 2000] que caracteriza la *diagnosticabilidad total*, dada anteriormente, y que se expresa como sigue: *Un Fallo 'F1' se dice que es discriminable de otro fallo 'F2' si y solo si existe alguna lectura de los sensores para la cual 'F1' aparece en algún candidato de diagnosis mínimo, pero 'F2' no aparece y viceversa.* Encontramos además definiciones para *grado de diagnosticabilidad* y *conjunto de sensores mínimos adicionales* para conseguir la diagnosticabilidad total.

De las definiciones dadas, se establece que para un determinado sistema, dos o más fallos no serán diagnosticables cuando, con la información de los sensores disponibles, no exista la posibilidad de establecer diferencias entre sus *trayectorias*. En la figura podemos observar dos grupos de *trayectorias* del mismo sistema, que se corresponden con fallos distintos del sistema, y que no son diferenciables entre sí, por tanto, no son diagnosticables.

Cuando esto ocurre, es obvio, que dar un diagnóstico resulta imposible. Por tanto, se plantea la posibilidad de dar todos los diagnósticos posibles cuando no se pueda precisar la causa de fallo de entre varias opciones. A continuación se plantean dos formas diferentes de afrontar este problema.

6.7.1. Etiquetado múltiple

La primera de las opciones que se plantean es actuar sobre la *base de datos etiquetada*. Una vez que se dispone de dicha base de datos, se recorre buscando *trayectorias* con etiquetas distintas que se encuentren muy próximas. Para definir la función de proximidad se usa una medida de distancia. Cuando se encuentran dos *trayectorias* pertenecientes a distintos comportamientos, que están dentro del umbral de proximidad, ambas *trayectorias* son etiquetadas con una nueva etiqueta,

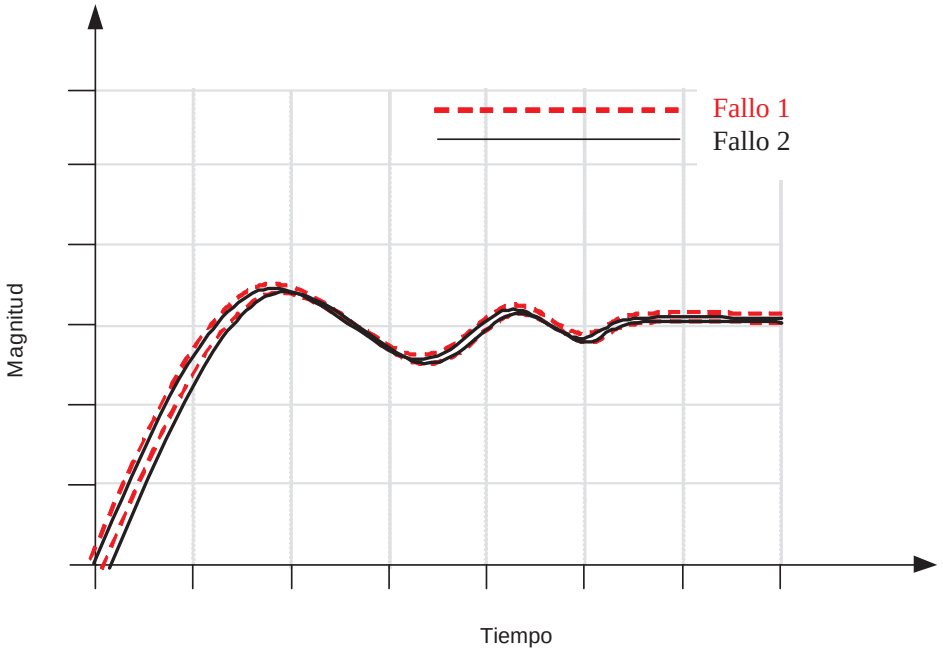

Figura 6.13: *Trayectorias* no diagnosticables entre si

que será la unión de las etiquetas de ambas. Se genera de esta forma una nueva clase para la futura clasificación, que corresponde a la unión de las clases originales.

Un concepto importante en el aprendizaje, es el concepto de similitud. La razón por la que usa este concepto es que, intuitivamente, datos similares tendrán clases o pertenecerán a grupos similares. Para medir la similitud se utiliza el concepto de *distancia*. Los métodos de similitud o de distancia se basan en almacenar los ejemplos analizados, y calcular la distancia del nuevo caso con el resto de ejemplos. Existe una variedad de funciones para calcular la distancia entre dos ejemplos:

- Euclídea

$$d(x,y) = \sqrt{\sum_{i=1}^{m}(x_i - y_i)^2}$$

- Minkowsky

$$d(x,y) = (\sum_{i=1}^{m} |x_i - y_i|^r)^{\frac{1}{r}}$$

- Manhattan

$$d(x,y) = \sum_{i=1}^{m} |x_i - y_i|$$

- Camberra

$$d(x,y) = \sum_{i=1}^{m} \frac{|x_i - y_i|}{x_i + y_i}$$

- Chebychev

$$d(x,y) = max_{i=1}^{m} |x_i - y_i|$$

Cuando aparecen atributos discretos la forma más simple consiste en definir la distancia para un atributo como 0 si los valores son iguales o 1 si son distintos. Una función de distancia alternativa para atributos discretos es VDM (*Value Difference Metric*) [Stanfill and Waltz, 1986]:

$$vdm_a(x,y) = \sum_{c=1}^{|\mathcal{C}|} (\frac{N_{a,x,c}}{N_{a,x}} - \frac{N_{a,y,c}}{N_{a,y}})^2$$

donde $N_{a,x}$ es el número de veces que el atributo a tiene el valor x, $N_{a,x,c}$ es el número de veces que teniendo el atributo a el valor x, la clase fue c, y $|\mathcal{C}|$ es el número de clases. Dos instancias son consideradas cercanas si tienen más clasificaciones similares, sin tener en cuenta el orden de los valores.

En caso de trabajar con atributos continuos y discretos, se puede utilizar la distancia heterogénea HVDM [Wilson and Martínez, 1997]:

$$HVDM(x,y) = \sqrt{\sum_{a=1}^{m} d_a(x_a - y_a)^2}$$

donde la función $d_a(x-y)$ es la distancia para el atributo a, definida como:

$$d_a(x-y) = \begin{cases} 1 & \text{si x o y son desconocido} \\ vdm_a(x,y) & \text{si a es nominal} \\ \frac{|x-y|}{4\sigma_a} & \text{si a es continuo} \end{cases}$$

donde $vdm_a(x,y)$ es la función dada y σ_a es la desviación estándar de los valores del atributo a en el conjunto de entrenamiento.

El algoritmo de etiquetado múltiple se expone en 6.4

Como puede apreciarse, el algoritmo recorre la *base de datos etiquetada*, comparando cada una de las *trayectorias* con todas las demás. Para cada par de *trayectorias*, se comparan primero sus etiquetas originales, es decir, aquellas etiquetas que tenían en la *base de datos etiquetada*. Si son distintas, se procede a

```
Algorithm Etiquetado_Multiple(umbral_proximidad)
  for i = 1 to total de trayectorias − 1 do
    Leer trayectoria i
    for j = i + 1 to total de trayectorias do
      Leer trayectoria j
      if etiqueta_original(i) ≠ etiqueta_original(j) then
        if distancia(trayectoria(i),trayectoria(j)) <umbral_proximidad then
          etiqueta(i) = etiqueta(i) ⋃ etiqueta_original(j)
          etiqueta(j) = etiqueta_original(i) ⋃ etiqueta(j)
        end if
      end if
    end for
  end for
```

Algoritmo **6.4:** Algoritmo de etiquetado múltiple

comprobar la distancia entre ambas. Si dicha distancia no supera el umbral que toma como parámetro el algoritmo, se entiende que dichas *trayectorias* no son diagnosticables entre sí, y se procede a su etiquetado múltiple. Para ello se añade al acumulado de etiquetas de cada *trayectoria*, la *trayectoria* original de la otra. Con esta forma de actuar, conseguimos que no se produzca la expansión de una etiqueta por transitividad.

De esta forma, la metodología debe ser modificada para realizar este paso antes de la fase de tratamiento. En la figura 6.14 se expone como quedaría la metodología con la incorporación del etiquetado múltiple de *trayectorias*.

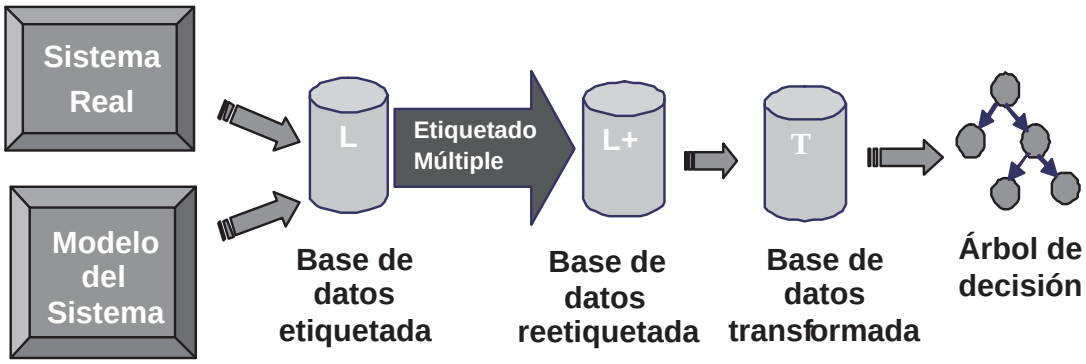

Figura 6.14: Inclusión del etiquetado múltiple en la metodología base

El problema que plantea el problema del etiquetado múltiple, es que necesitamos definir la función de distancia y el umbral de proximidad para poder realizarlo.

6.7.2. Clasificación binaria

Otra forma de obtener más de un posible diagnóstico cuando los fallos producen *trayectorias* muy similares es actuar sobre la generación de los árboles de clasificación. Implícitamente, también lleva un cambio en las etiquetas de la *base de datos etiquetada*, pero no en el mismo sentido que antes.

La idea es que, si dos *trayectorias* están muy próximas entre sí, y pertenecen a comportamientos diferentes, un sólo clasificador errará al clasificar una de ellas, ofreciéndo un único diagnóstico. Pero, si tenemos un clasificador específico para cada clase, ambos clasificadores catalogarán cada una de dichas *trayectorias* como pertenecientes a su clase, con lo cual, si se hace la evaluación de los dos en paralelo tendremos ambos diagnósticos. Con esto conseguimos el mismo efecto que en el caso del etiquetado múltiple (sección 6.7.1), es decir, más de un posible diagnóstico cuando las *trayectorias* no son diagnosticables unívocamente.

La ventaja que ofrece esta forma de actuar, es que no se necesita establecer un criterio de distancia, ni un umbral de proximidad, es el propio clasificador el que establece la pertenencia o no de la clase.

Así pues necesitamos convertir el problema de clasificación múltiple que tenemos en un problema de clasificación binaria.

Cuando tenemos un problema de clasificación de más de dos clases y queremos resolverlo mediante clasificación que sólo consiga discriminar entre dos clases (clasificación binaria) lo que estamos haciendo es conocido como binarización. Existen varios métodos para realizar la binarización:

- Uno contra el resto. Se construye un clasificador binario usando todos los ejemplos de una clase y agrupando en una misma clase el resto de ejemplos. Esto se realiza para todas las clases. Si existen n clases, el resultado serán n clasificadores binarios que posteriormente habrá que combinar.

- Todos los pares. Se construye un clasificador binario utilizando todos los ejemplos de dos clases e ignorando el resto. Esto se realiza para todos los

pares de clases. Si existen n clases, el resultado serán $n(n-1)/2$ clasificadores binarios que posteriormente habrá que combinar.

- Todas las mitades. Se construye un clasificador binario utilizando los ejemplos de la mitad de las clases por un lado y el resto por el otro. Si tenemos n clases y el número es par tendremos $n/2$ clases en cada mitad. Esto se realiza para todas las particiones posibles del número de clases. La combinación en este caso es más costosa, puesto que ninguno de los clasificadores binarios representa a clases simples.

En el caso que nos ocupa necesitamos recurrir a la clasificación de *uno contra el resto*. Para llevar a cabo esta idea, necesitamos actuar nuevamente sobre la *base de datos etiquetada*. En este caso, dicha base de datos deberá ser replicada tantas veces como etiquetas diferentes existan.

Cada una de las bases de datos resultantes conservará una única etiqueta de la *trayectoria* original, y modificará el resto para que indiquen el opuesto a la etiqueta conservada. De esta forma tendremos una *base de datos etiquetada* con dos únicas etiquetas, el comportamiento conservado, y la negación del comportamiento conservado.

La metodología base se aplicará a cada una de las nuevas bases de datos, con lo que finalmente obtendremos un árbol de decisión para cada una de ellas. Cada uno de estos árboles de decisión tan sólo será capaz de establecer la pertenencia o no de una nueva observación al comportamiento que evalúa. Para conseguir el diagnóstico se evaluarán todos los árboles de decisión obtenidos.

En la figura 6.15 se representa como quedaría la fase *off-line* de la metodología con esta modificación.

Por supuesto, para que la metodología funcione correctamente, el clasificador debería priorizar la clasificación correcta de la clase que conserva la etiqueta original. Es decir, los errores en la clasificación, deberán producirse para la clase opuesta a la original. De ser así, dos *trayectorias* con etiquetas distintas pero que sean iguales, serán clasificadas como pertenecientes a la clase de la etiqueta original, produciendo errores en la clasificación de la otra.

La fase *on-line* de la metodología deberá acomodarse a la nueva situación, evaluando la nueva observación con cada uno de los árboles obtenidos. En la figura 6.16 se puede observar dicha situación.

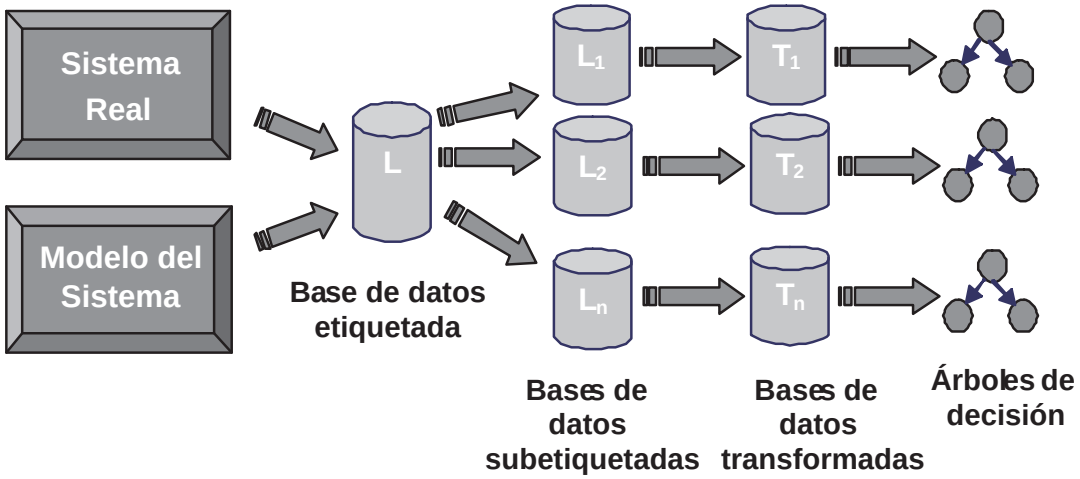

Figura 6.15: Inclusión de la clasificación binaria en la metodología base

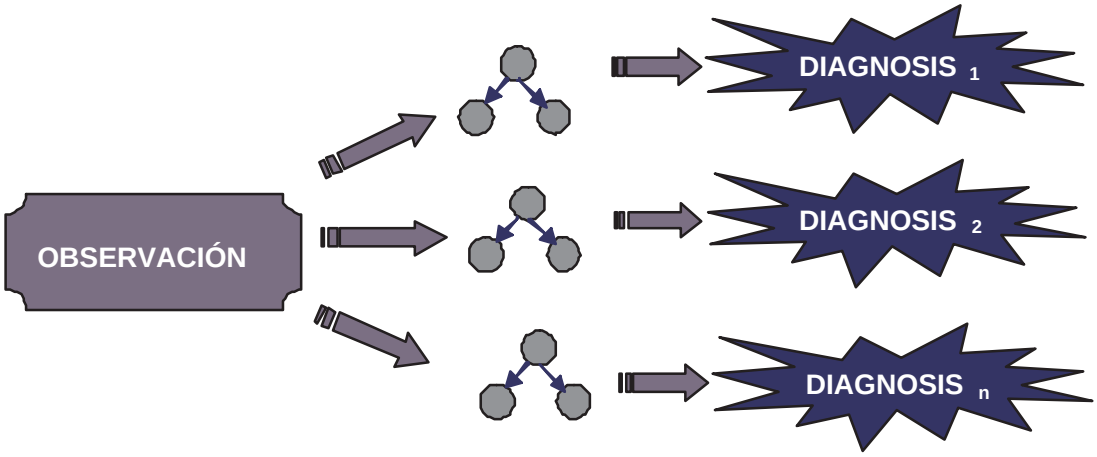

Figura 6.16: Evaluación de la clasificación binaria

Ante esta perspectiva, las posibilidades que se nos ofrecen de diagnóstico son:

1. Sólo un árbol proporciona una diagnosis positiva. Esto significa que el diagnóstico será único.

2. Varios árboles proporcionan una diagnosis positiva. Esto significa que la observación puede corresponderse a más de un comportamiento, y los posibles comportamientos son los ofrecidos, sin poder precisar cual es el verdadero.

3. Ninguno de los árboles ofrece un diagnóstico positivo. En tal caso, la observación no puede ser evaluada, y deberá catalogarse como un fallo para que el sistema de diagnóstico no ha sido preparado, es decir, el resultado del diagnóstico será *fallo desconocido*.

6.8. DISETRA: un algoritmo para la clasificación de *trayectorias*

De los ensayos realizados con la metodología presentada en las secciones anteriores, se deduce que muchas veces el error del diagnóstico viene dado por la colocación de la frontera que realiza el clasificador a la hora de separar dos clases.

Se deduce también, que para que dos conjuntos de *trayectorias* sean correctamente clasificadas, tan sólo se necesita que alguno de los atributos con los que se caracterizan las diferentes clases de *trayectorias* pertenezcan a conjuntos disjuntos. Si esto ocurre, es posible encontrar una regla que clasifique dichos conjuntos de *trayectorias*.

El algoritmo que se presenta en esta sección es el resultado de estas observaciones. Este algoritmo pretende encontrar un atributo de los conjuntos de *trayectorias* para el cual todos sus valores pertenezcan a conjuntos disjuntos. De ser así, considerará dichos conjuntos de *trayectorias* como separables y generará una regla de clasificación que intentará maximizar la separación de dichas clases. El algoritmo incorpora como parte de sí mismo las técnicas de tratamiento de *trayectorias* de la metodología que han demostrado ser válidas para la clasificación de las clases.

DISETRA es un algoritmo de clasificación que pretende encontrar un conjunto de reglas con las que clasificar una *base de datos etiquetada*. Este conjunto de reglas

permite evaluar una nueva observación con el objetivo de llegar al diagnóstico del sistema.

Para poder explicar el funcionamiento de DISETRA necesitamos definir los siguientes conceptos:

Definición 6.13. **Conjunto de trayectorias**. Es el conjunto de *trayectorias etiquetadas* pertenecientes al mismo comportamiento, es decir, que tienen la misma etiqueta.

Definición 6.14. **Conjuntos de trayectorias separables**. Dos *conjuntos de trayectorias* CA y CB se dice que son *separables* si ocurre alguna de las siguientes situaciones:

1. Existe al menos un instante de tiempo i, tal que, el valor de cualquier trayectoria de CA en i es mayor que el valor de cualquier trayectoria de CB en i, o el valor de cualquier trayectoria de CA en i es menor que el valor de cualquier trayectoria de CB en i. En ese caso, dichos conjuntos son *separables* en el instante i.

$$\exists i \in \{1..k\} | (\forall j_a(i) \in CA, \forall j_b(i) \in CB \Rightarrow j_a(i) > j_b(i) \vee \forall j_a(i) \in CA, \forall j_b(i) \in CB \Rightarrow j_a(i) < j_b(i))$$

2. Existe al menos un instante de tiempo i, tal que, la derivada de cualquier trayectoria de CA en i es mayor que la derivada de cualquier trayectoria de CB en i, o la derivada de cualquier trayectoria de CA en i es menor que la derivada de cualquier trayectoria de CB en i. En ese caso, dichos conjuntos son *separables* en el instante i.

$$\exists i \in \{1..k\} | (\forall j_a(i) \in CA, \forall j_b(i) \in CB \Rightarrow \dot{j_a}(i) > \dot{j_b}(i) \vee \forall j_a(i) \in CA, \forall j_b(i) \in CB \Rightarrow \dot{j_a}(i) < \dot{j_b}(i))$$

   La figura 6.17 muestra dos *conjuntos de trayectorias* separables con este criterio

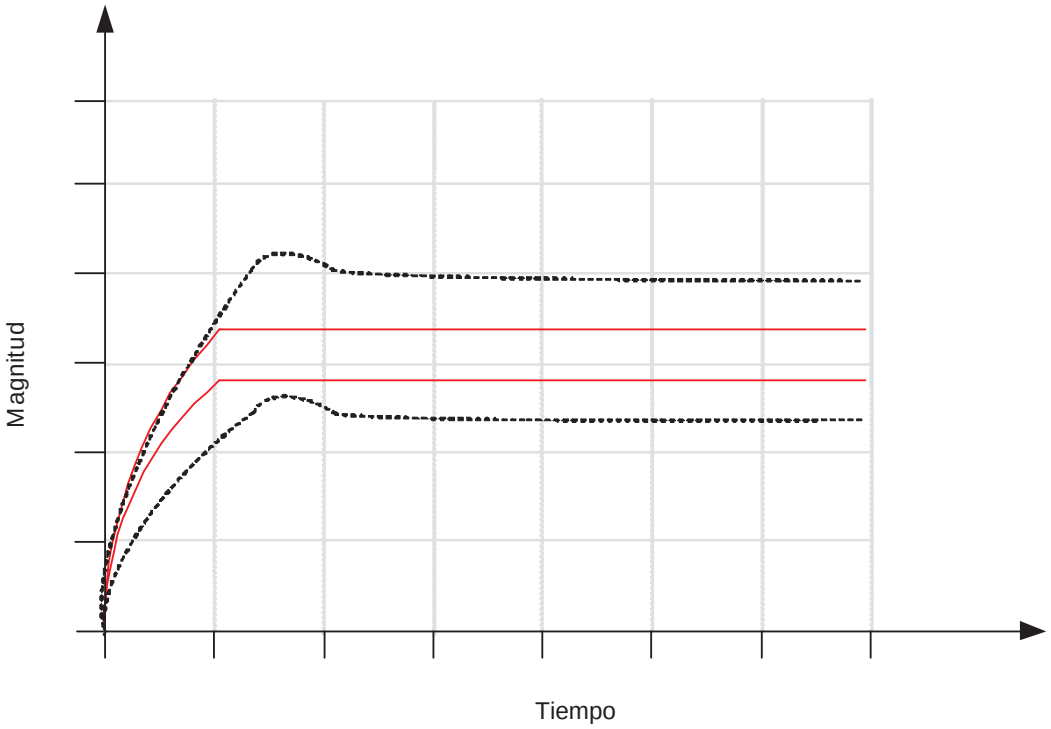

Figura 6.17: Conjuntos de trayectorias separables

3. La suma de cambios de signo de las derivadas de cualquier *trayectoria* de CA es mayor que la suma de los cambios de signo de las derivadas de cualquier *trayectoria* de CB, o la suma de cambios de signo de las derivadas de cualquier *trayectoria* de CA es menor que la suma de los cambios de signo de las derivadas de cualquier *trayectoria* de CB.

$$\forall j_a \in CA, \forall j_b \in CB \Rightarrow \breve{j}_a > \breve{j}_b \vee \forall j_a \in CA, \forall j_b \in CB \Rightarrow \breve{j}_a < \breve{j}_b$$

siendo $\breve{j}_i$, el número de cambios de signo de la derivada en una *trayectoria* j del conjunto de trayectorias CI. La figura 6.18 muestra dos *conjuntos de trayectorias* separables con este criterio

El algoritmo DISETRA busca los *conjuntos de trayectorias separables* en una *base de datos etiquetada*. Para ello sigue los siguientes pasos:

Paso 1: Segmentación de la *base de datos etiquetada*. Para estudiar la *separabilidad* de los *conjuntos de trayectorias* es necesario disponer de las

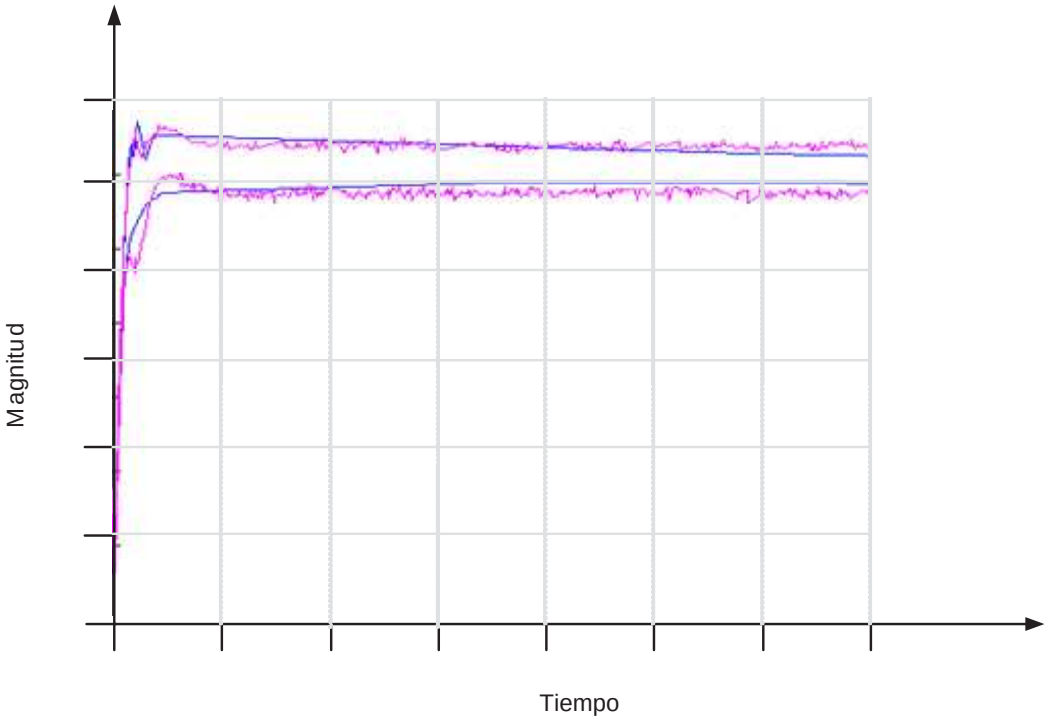

Figura 6.18: Conjuntos de trayectorias separables

derivadas de cada uno de los puntos de las *trayectorias*, además de calcular el número de cambios de signo que dichas derivadas. Toda esta información deberá añadirse a la ya existente en la *trayectoria etiquetada*. Esto genera dos problemas:

- Al añadir los valores de las derivadas de cada instante de la *trayectoria*, el número de elementos de la *trayectoria* se duplica, haciendo mayor la, ya de por si grande, *trayectoria*.
- La presencia habitual de ruido en las *trayectorias* hace que el cálculo de la derivada en un punto no sea representativa de la derivada real de la *trayectoria* en dicho punto.

Para evitar dichos problemas se procederá a la segmentación de cada una de las *trayectorias* presentes en la *base de datos etiquetada*. Para la segmentación se emplea el algoritmo descrito en el apartado 6.4.4.

Una vez segmentada la *base de datos de trayectorias*, se procede a calcular e incluir, para cada *trayectoria*, la pendiente de cada segmento y el número total de cambios de signo de las pendientes.

El resultado de este paso será una *base de datos de trayectorias segmentadas y etiquetadas*.

Paso 2: Normalización de la *base de datos de trayectorias segmentadas*. El proceso de segmentación ha aproximado cada *trayectoria* mediante un conjunto se segmentos, pero la base de datos resultante no cumple los requisitos de una *base de datos etiquetada*, puesto que no todas las *trayectorias* tienen el mismo número de elementos. Cada *trayectoria* ha sido representada con un número de segmentos distintos, que además comienzan y terminan en instantes de tiempo diferentes.

Para solucionar esto, se deberán generar nuevos segmentos que igualen el número de segmentos de cada *trayectoria* y los instantes de tiempo de dichos segmentos. Este proceso será realizado mediante el algoritmo de normalización mostrado en la sección 6.4.4.

El resultado de este paso será una *base de datos de trayectorias segmentadas y normalizadas*.

Paso 3: Búsqueda de conjuntos de trayectorias totalmente separables. Un *conjunto de trayectorias* se dice que es *totalmente separable* si es *separable* de todas y cada una del resto de *conjuntos de trayectorias*. En este paso se evalúa la *separabilidad* de cada uno de los *conjuntos de trayectorias* con el resto. Este proceso se realizará para cada uno de los *conjuntos de trayectorias* de la *base de datos de trayectorias segmentadas y normalizadas*.

Para cada *conjunto de trayectorias totalmente separable* se genera una regla que permita clasificar a dicho conjunto de *trayectorias*.

En 6.5 se muestra el algoritmo de búsqueda de los conjuntos de trayectorias totalmente separables.

Paso 4: Búsqueda de conjuntos de trayectorias separables individualmente y no separables. En este paso se comprueba la *separabilidad* por parejas de los *conjuntos de trayectorias* que no son *totalmente separables*.

```
Algorithm Totalmente_Separables(Conjuntos de trayectorias)
  Calcular máximos y mínimos de cada conjunto de trayectorias
  for i = 1 to total de conjuntos de trayectorias do
    for t = 1 to total de instantes de tiempo do
      if max(conjunto(i)) < todos los mínimos del resto de conjuntos then
        conjunto(i) es separable totalmente en el instante t
      else if min(conjunto(i)) > todos los máximos del resto de conjuntos then
        conjunto(i) es totalmente separable en el instante t
      end if
    end for
  end for
```

Algoritmo **6.5:** Algoritmo de búsqueda de conjuntos de trayectorias totalmente separables

Por cada *conjunto de trayectorias separable* de otros, se genera una regla que clasifique dicho comportamiento.

Finalmente, aquellos conjuntos de reglas que no sea posible separar de otros generan una regla final, donde la clasificación resultante sea la combinación de todos los conjuntos de *trayectorias no separables*.

En 6.6 se muestra el algoritmo de búsqueda de los conjuntos de trayectorias separables inidvidualmente y no separables.

Selección del instante y el valor de clasificación. Una vez que se ha comprobado que un *conjunto de trayectorias* es *separable* de otro, es posible que tengamos muchos instantes de tiempo en los cuales los *conjuntos de trayectorias* sean *separables*. Lógicamente, para la generación de reglas sólo nos podemos quedar con uno de ellos. Llegados a este punto tenemos diversas posibilidades para seleccionar el instante para el que generamos la regla:

a) Se selecciona el primero de los instantes. Esto permite abortar el algoritmo de búsqueda de *separabilidad* y es muy apropiado para la diagnosis, puesto que permite un diagnóstico más temprano.

b) Se selecciona el instante de máxima separación. Será el instante de tiempo en el que los *conjuntos de trayectorias* están más separados. Con

```
Algorithm Separables_Individualmente(Conjuntos de trayectorias)
  Calcular máximos y mínimos de cada conjunto de trayectorias
  m= total de conjuntos no totalmente separables
  for i=1 to m-1 do
    for j=2 to m do
      for t=1 to total de instantes do
        if max(conjunto(i)) <min(conjunto(j)) then
          conjunto(i) y conjunto(j) separables individualmente en instante t
        else if min(conjunto(i)) >max(conjunto(j)) then
          conjunto(i) y conjunto(j) separables individualmente en instante t
        end if
      end for
    end for
  end for
```

Algoritmo **6.6:** Algoritmo de búsqueda de conjuntos de trayectorias separables individualmente y no separables

esta opción se consigue ser menos sensible al conjunto de *trayectorias* de ejemplo, puesto que al ser la separación máxima, se prevé que cualquier otra *trayectoria* no presente en el conjunto de ejemplo, pero distante de ellas a pesar de tener la misma clase, tiene más margen para ser correctamente clasificada.

c) Se selecciona el instante central del intervalo más amplio durante el cual son *separables*. Esto garantiza que la regla soporte bien posibles retardos o adelantos de las *trayectorias*.

En la figura 6.19 se muestran gráficamente cada una de las tres posibilidades.

Podríamos también, realizar una evaluación ponderada de todos ellos para seleccionar el instante óptimo con el que generar la regla.

Una vez seleccionado el instante de tiempo para el cual se va a generar la regla, el valor de clasificación será el punto medio del intervalo que separa a las dos *trayectorias* más próximas en dicho instante. Es decir, un punto equidistante de las dos *trayectorias* más próximas de comportamientos distintos.

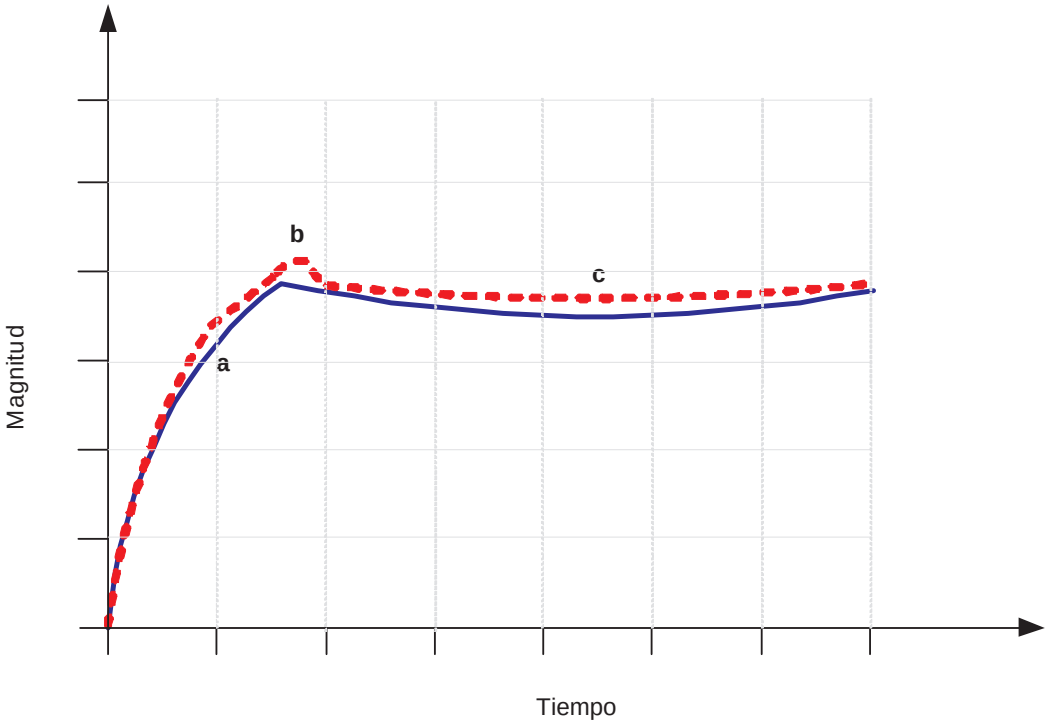

Figura 6.19: Selección del instante y valor de clasificación

Capítulo 7

Casos de Estudio

7.1. Introducción

Los motores eléctricos son componentes muy comunes dentro de los procesos industriales debido a su robustez, coste limitado y facilidad de mantenimiento. Es habitual que este tipo de motores formen parte de sistemas industriales más complejos. La detección de fallos y la diagnosis de estos motores es muy importante cuando estos trabajan en condiciones de monitorización *on-line*. Debido a las condiciones dinámicas de los sistemas, es crucial que la diagnosis de los fallos se realice de forma rápida y precisa. Se han aplicado multitud de técnicas a la diagnosis de este tipo de motores: técnicas basadas en el análisis de señales [Schoen *et al.*, 1995]; basadas en el modelado dinámico del motor [Chan *et al.*, 1999]; basadas en la lógica borrosa [de Miguel and Blazquez, 2005; Sainz Palmero *et al.*, 2005] y por supuesto, técnicas basadas en conocimiento. Dentro de éstas últimas, se han usado muchas aproximaciones: sistemas expertos [Filippetti *et al.*, 1992], redes neuronales [Liu *et al.*, 2000] y aprendizaje automático [Hajiaghajani *et al.*, 2004; Casimir *et al.*, 2006].

Es habitual que estos motores trabajen en modo iterativo, repitiendo continuamente una serie de cambios de regimen de giro entre unos pocos puntos de establecimiento. Además, sería deseable no tener que añadir sensores adicionales al motor para incorporarle el sistema de diagnóstico de fallos, sino ser capaces de diagnosticar únicamente con la información disponible de los sensores presentes. Por supuesto, uno de dichos sensores si se pretende que el motor gire a un re-

gimen determinado será el sensor de velocidad de giro. Este será el caso de los experimentos mostrados en el presente capítulo.

El objetivo final será diagnosticar los fallos de un motor de corriente continua que varía de regimen de giro entre referencias conocidas. El sistema dispondrá de un sistema de control que permitirá alcanzar la referencia. Con ese propósito se emplearán las técnicas expuestas en el capítulo 6. Para demostrar la validez de dichas técnicas, se comenzará estudiando un modelo simple del sistema a diagnosticar, para luego probar con un modelo detallado y fallos más reales y específicos; y finalmente demostrar la validez de la metodología con un sistema real.

7.2. Modelo básico de lazo de control de un motor eléctrico

7.2.1. Descripción del Sistema

Como primer problema de sistema dinámico cuyo comportamiento debe ser diagnosticado se ha elegido el presentado en [Panati *et al.*, 2000], que es una modificación del presentado en un trabajo previo [Malik and Struss, 1996].

El modelo de sistema dinámico considerado se representa en la Figura 7.1. Consta de un motor eléctrico (M), cuya velocidad angular (W) se controla mediante el voltaje (V) proporcionado por el controlador (C). El controlador (C) actúa basándose en la velocidad angular deseada (d) y el valor de la velocidad angular medida (W_m) por el sensor (S).

En [Malik and Struss, 1996] se considera un controlador proporcional (P) para el modelo, mientras que en el trabajo de [Panati *et al.*, 2000] se considera un sistema dinámico con un controlador proporcional integral (PI), que será el mismo que consideraremos para nuestro ejemplo.

Las ecuaciones que caracterizan el comportamiento de cada uno de los componentes son: la ecuación 7.1 para el comportamiento del motor, la ecuación 7.2 para el comportamiento del controlador y la ecuación 7.3 para el comportamiento del sensor. C_m, C_c y C_s son las constantes que rigen al motor, al sensor controlador y al sensor respectivamente.

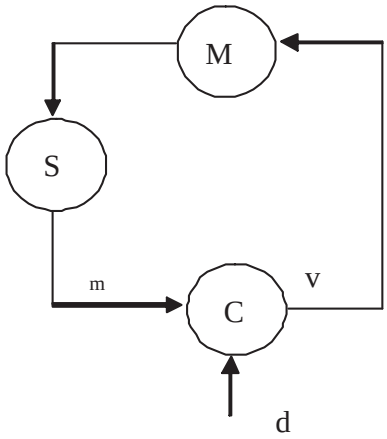

Figura 7.1: Lazo de Control de un motor eléctrico

$$T \cdot \frac{d\omega}{dt} = c_m \cdot v - \omega \tag{7.1}$$

$$\frac{d\omega}{dt} = C_c \cdot (d - \omega_m) \tag{7.2}$$

$$\omega_m = C_s \cdot \omega \tag{7.3}$$

Inicialmente partiremos de una descripción del problema, que para nuestro caso será el presentado con anterioridad. A partir de dicha descripción debemos modelar dicho sistema para poder comenzar a realizar simulaciones con él. En el caso que nos ocupa el sistema se ha modelado como un diagrama de Forrester [Forrester, 1968], para poder realizar simulaciones usando la herramienta de simulación VEMSIM®. El diagrama de Forrester generado para el sistema de ejemplo que nos ocupa es el que se presenta en la figura 7.2. Las variables f y $F2$ del diagrama de Forrester son variables de flujo, y la función $INTEG$ representa la integral. De esta forma representamos formalmente el sistema de ejemplo de la figura 7.1.

Las ecuaciones que relacionan las diferentes variables que aparecen en el diagrama son las que se muestran en 7.4 hasta 7.8:

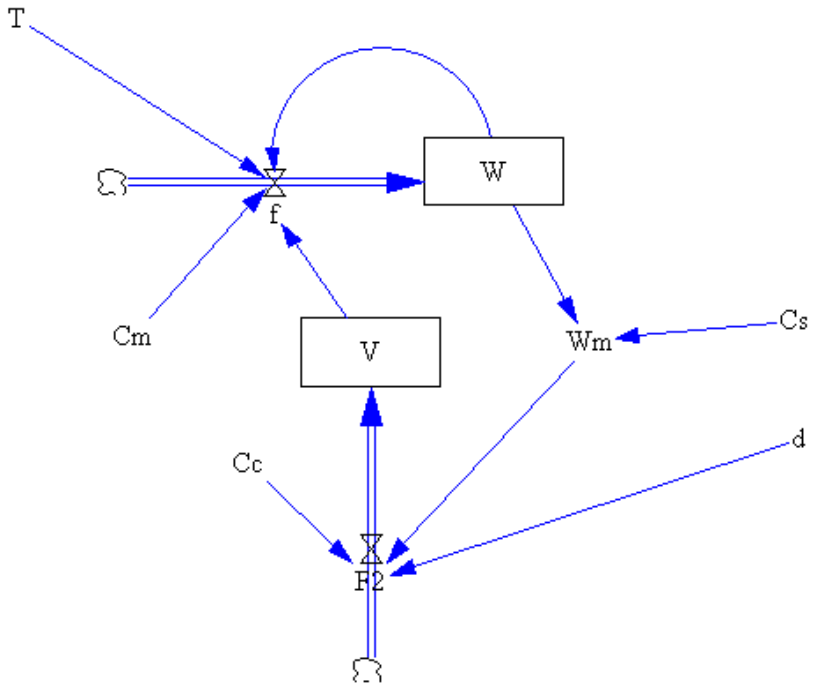

Figura 7.2: Modelado mediante diagrama de Forrester

T	3
d	10
W	5
Cm	[0.98 - 1.02]
Cc	[0.98 - 1.02]
Cs	[0.98 - 1.02]

Tabla 7.1: Valores correctos

$$W_m = W \cdot C_s \tag{7.4}$$

$$V = INTEG(f2) \tag{7.5}$$

$$W = INTEG(f) \tag{7.6}$$

$$f2 = C_c \cdot (d - W_m) \tag{7.7}$$

$$f = \frac{C_m \cdot V - W}{T} \tag{7.8}$$

Se considerará que los valores correctos para el sistema serán los de la tabla 7.1:

Con esto suponemos que nuestro sistema tiene un comportamiento correcto mientras que los valores de las constantes de cada componente oscilen en el rango $[0{,}98 - 1{,}02]$, y que la situación de estudio se corresponde con el motor subiendo de 5 rad/seg a 10 rad/seg, teniendo el motor elegido una inercia (T) igual a 3.

Como podemos ver las constantes de los distintos componentes tienen un rango de tolerancia, lo cual simula los posibles ruidos y pequeñas desviaciones que se producen en los sistemas reales, mientras que consideramos que la inercia del motor es fija y tiene un valor de 3. Los datos de la velocidad angular medida W_m se obtendrán cada 0,1 unidades de tiempo, que es el tiempo de muestreo considerado para este caso.

7.2.2. Identificación de fallos

Los posibles fallos que podemos encontrarnos en nuestro sistema son provocados por el funcionamiento anómalo de alguno de los componentes. Esto se debe principalmente a la desviación del valor nominal de las constantes relacionadas con el componente. Al efectuar la diagnosis, pues, no sólo se deberá indicar el componente que está fallando, sino también si la constante que rige dicho componente se sitúa en valores por encima o por debajo del rango correcto. Las posibles causas de fallos que deseamos identificar son por tanto:

- La constante del motor se encuentra por encima o por debajo de los valores correctos. Para el caso de que la constante esté por encima del rango correcto, el fallo provocado se denotará por **Cm Alta**. Por el contrario, si la constante se encuentra por debajo del rango correcto el fallo provocado se denotará pro **Cm Baja**.

- La constante del controlador se encuentra por encima o por debajo de los valores correctos. Para el caso de que la constante esté por encima del rango correcto, el fallo provocado se denotará por **Cc Alta**. Por el contrario, si la constante se encuentra por debajo del rango correcto el fallo provocado se denotará pro **Cc Baja**.

- La constante del sensor se encuentra por encima o por debajo de los valores correctos. Para el caso de que la constante esté por encima del rango correcto, el fallo provocado se denotará por **Cs Alta**. Por el contrario, si la constante

se encuentra por debajo del rango correcto el fallo provocado se denotará pro **Cs Baja**.

Para el funcionamiento correcto, los valores de las constantes se podrán mover en un rango de valores que denominamos correctos, puesto que todo componente real tiene un cierto margen de tolerancia, de esta forma nos acercamos lo más posible en nuestra simulación a la realidad. El comportamiento correcto será denotado por **OK**.

Esto provoca que nuestro sistema tendrá una familia de comportamientos correctos, correspondientes a todas las combinaciones posibles de las constantes de los componentes que se encuentren dentro de los límites de tolerancia considerados. Esto es una dificultad añadida a la hora de realizar la diagnosis, al no tener un valor de referencia fijo y unívoco que represente un comportamiento correcto sobre el que medir las posibles desviaciones que se puedan producir, pero por el contrario proporciona una visión más realista del sistema.

Las figuras 7.3, 7.4 y 7.5 muestran los comportamientos OK, Cm Alta y Cc Baja respectivamente.

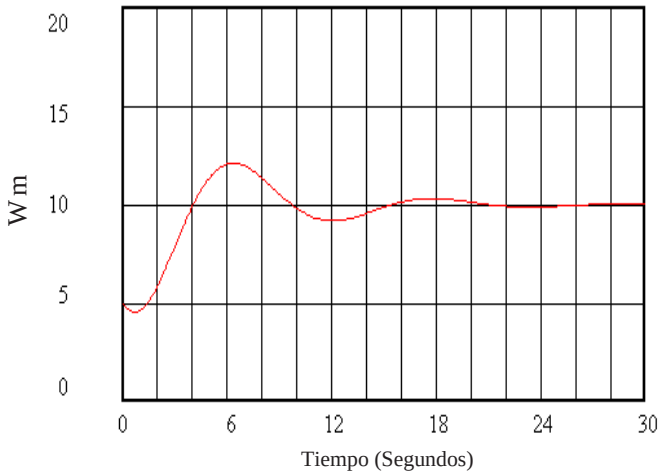

Figura 7.3: Comportamiento OK

En la figura 7.6 se muestra una comparativa de dos comportamientos distintos con evoluciones muy similares. En concreto, los comportamientos mostrados son *Cc Alta* versus *Cm Alta*. Este tipo de situaciones, donde dos fallos distintos provocan comportamientos del sistema similares son los más difíciles de diagnosticar.

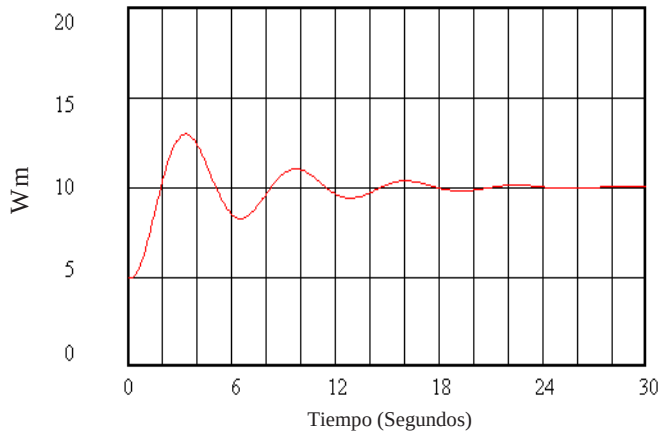

Figura 7.4: Comportamiento Cm Alta

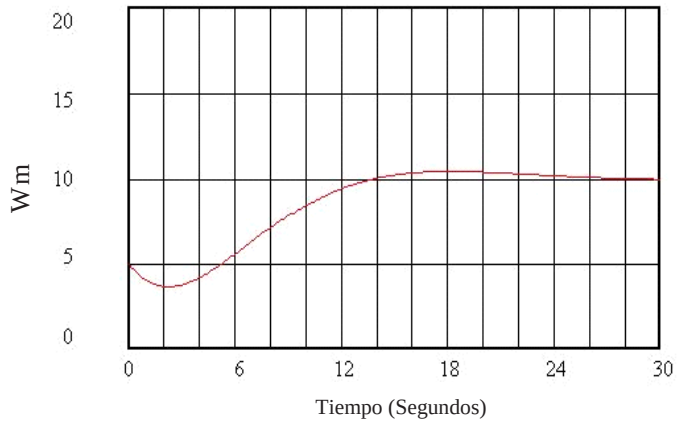

Figura 7.5: Comportamiento Cc Baja

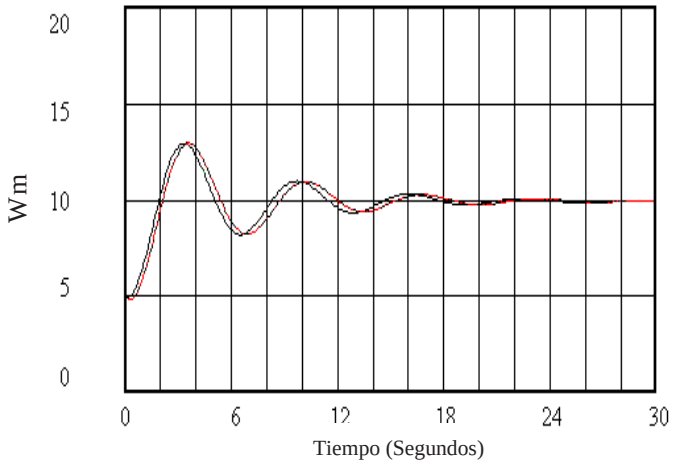

Figura 7.6: Comportamiento Cc Alta vs Cm Alta

Fase *off-line*

Para nuestro ejemplo vamos a realizar 100 simulaciones para valores de las constantes de los componentes dentro de su rango correcto. Estas 100 simulaciones que vamos a obtener se etiquetarán como OK, y representan la familia de comportamientos correctos para nuestro sistema.

De la misma forma se genera el conjunto de simulaciones (100 para cada caso) correspondientes a las etiquetas CmAlta, CmBaja, CcAlta, CcBaja, CsAlta y CsBaja, donde los valores de las constantes variarán entre [0.1-0.97] para los valores en los que estén bajas y [1.03-5] para los valores en los que estén altas. Cada una de estas simulaciones se etiquetará a continuación con la etiqueta correspondiente a su comportamiento. Obteniendo por tanto un total de 700 simulaciones.

Una vez generada la base de datos de simulaciones debemos proceder al tratamiento de esa información. Este tratamiento vendrá dado en dos sentidos, en primer lugar la información recogida por la variables W_m es muy pobre como para diagnosticar posibles comportamientos futuros, y en segundo lugar la cantidad de puntos que forman una trayectoria es excesivamente grande como para poder trabajar con ellos. Por tanto el tratamiento de los datos consistirá en reducir el número de puntos que forman la trayectoria y enriquecerlos con más información.

Para reducir el número de puntos que forman la trayectoria realizamos un muestreo de la misma. Es necesario seleccionar los puntos que forman la muestra de forma constante, puesto que cualquier otro método elegido, por ejemplo un método estadístico, modificaría la forma de la curva, base para nuestra diagnosis. El único motivo para reducir el número de puntos de cada trayectoria es la imposibilidad de manejar trayectorias excesivamente grandes por parte del clasificador. Mientras menos puntos tenga la muestra más fácil será de trabajar con ella, pero por el contrario perdemos parte de la información sobre la trayectoria. En nuestro caso el muestreo se realiza tomando uno de cada tres puntos de la trayectoria original.

Para cada uno de esos puntos seleccionados se calculan las siguientes magnitudes, con objeto de obtener mayor información del comportamiento de la trayectoria de la que proporcionan únicamente las magnitudes medidas.

1. *Inclinación relativa* (IR).

2. *Distancia al comportamiento perfecto* (DP).

3. *Inclinación relativa a la trayectoria perfecta*(IRP).

La definición de dichas magnitudes fue expuesta en la sección 6.4.5. De esta forma tendremos que hemos pasado de una base de datos de trayectorias donde cada trayectoria se representa de la forma:

$$W_m[1], W_m[2], W_m[3], ...W_m[n], ..., ETIQUETA$$

A otra donde cada trayectoria se representa de la forma:

$$W_m[i], IR(i), DP(i), IRP(i), W_m[j], IR(j), DP(j), IRP(j), ..., ETIQUETA$$

Esta base de datos servirá como entrada para realizar el aprendizaje supervisado. El aprendizaje se llevará a cabo mediante una herramienta de clasificación, que en nuestro caso será el C4.5 [Quinlan, 1993a], que será capaz de clasificar, en función de la etiqueta, los valores que deben tomar cada uno de los datos aportados para que se correspondan con el comportamiento asociado a la etiqueta. Lo que se consigue con esta herramienta es caracterizar cada una de las familias de comportamiento según los valores de los atributos que se les ha proporcionado. Para los datos de nuestro ejemplo el tamaño del clasificador generado es de 37 nodos.

Resultados

Para evaluar la metodología vamos a llevar a cabo un conjunto de pruebas que nos permitan ver lo acertado de las diagnosis obtenidas.

Los datos de observación del sistema se obtienen realizando la simulación del mismo en diferentes condiciones de funcionamiento. De esta forma conocemos la diagnosis que deberíamos obtener y lo podemos comparar con la diagnosis obtenida a partir del conjunto de reglas. Por supuesto, para que sean evaluables, las nuevas observaciones son tratadas en la misma forma en que lo fueron los datos para el entrenamiento, es decir, se muestrea uno de cada tres observaciones, y se añaden los atributos calculados descritos anteriormente.

El resultado de la diagnosis de este caso se puede ver en la siguiente tabla de pruebas:

Observamos que en los casos en los que los fallos vienen dados por valores de las constantes muy cercanas a la de los valores correctos, la diagnosis no es acertada,

Valores de funcionamiento			Diagnosis Correcta	Diagnosis Obtenida
Cm	**Cc**	**Cs**		
1	1	1.03	Cs Alta	OK
1	1	1.07	Cs Alta	Cs Alta
1	1	1.1	Cs Alta	Cs Alta
1	1	1.5	Cs Alta	Cs Alta
1	1	2	Cs Alta	Cs Alta
1	1	3	Cs Alta	Cs Alta
1	1.03	1	Cc Alta	OK
1	1.07	1	Cc Alta	Cc Alta
1	1.1	1	Cc Alta	Cc Alta
1	1.5	1	Cc Alta	Cc Alta
1	2	1	Cc Alta	Cc Alta
1	3	1	Cc Alta	Cc Alta
1.03	1	1	Cm Alta	OK
1.07	1	1	Cm Alta	Cc Alta
1.1	1	1	Cm Alta	Cc Alta
1.5	1	1	Cm Alta	Cm Alta
2	1	1	Cm Alta	Cm Alta
3	1	1	Cm Alta	Cm Alta
1	1	0.97	Cs Baja	Cs Baja
1	1	0.93	Cs Baja	Cs Baja
1	1	0.89	Cs Baja	Cs Baja
1	1	0.85	Cs Baja	Cs Baja
1	1	0.5	Cs Baja	Cs Baja
1	1	0.1	Cs Baja	Cs Baja
1	0.97	1	Cc Baja	OK
1	0.93	1	Cc Baja	Cc Baja
1	0.89	1	Cc Baja	Cc Baja
1	0.85	1	Cc Baja	Cc Baja
1	0.5	1	Cc Baja	Cc Baja
1	0.1	1	Cc Baja	Cc Baja

Valores de funcionamiento			**Diagnosis Correcta**	**Diagnosis Obtenida**
Cm	**Cc**	**Cs**		
0.97	1	1	Cm Baja	OK
0.93	1	1	Cm Baja	Cm Baja
0.89	1	1	Cm Baja	Cm Baja
0.85	1	1	Cm Baja	Cm Baja
0.5	1	1	Cm Baja	Cm Baja
0.1	1	1	Cm Baja	Cm Baja
0.99	0.98	1.02	OK	OK
1	1.02	1.02	OK	OK
0.98	1	0.98	OK	OK
0.98	1.02	1.02	OK	OK
0.99	1.01	1.01	OK	OK
1.01	1	0.99	OK	OK

Tabla 7.2: Resultados obtenidos con la metodología base

indicando que el comportamiento es correcto. Esto se debe a la relación existente entre las constantes, y a permitir un rango de valores para los comportamientos correctos, lo que provoca que esa pequeña desviación en una de las constantes pueda ser 'compensado' por un valor de otra de las constantes en sentido contrario pero dentro de sus márgenes de tolerancia.

La diagnosis también es errónea para ciertos valores de *CmAlta*, diagnosticándose *CcAlta*. Esto se debe a que la estrecha relación existente entre las constantes de los componentes para este sistema provoca que alteraciones complementarias en estas constantes haga que el sistema evolucione de una misma forma. En la figura 7 podemos ver como para los mismos valores de Cs y con valores de Cm y Cc complementados el comportamiento del sistema es prácticamente idéntico. En este caso las simulaciones se han realizado con $Cs = 1$, $Cc = 1$ y $Cm = 3$ para *CmAlta* y $Cs = 1$, $Cc = 2{,}8$ y $Cm = 1$ para *CcAlta*.

En general la diagnosis es bastante acertada, sobre todo cuando la constante que falla se aleja significativamente de sus valores correctos.

Hay que resaltar también que mientras los valores de las constantes se mantienen en el rango que se considera de funcionamiento correcto, la metodología

propuesta funciona correctamente, aunque los valores de dichas constantes estén en sus extremos.

Para el juego de pruebas que se ha presentado se consigue diagnosticar el 80.55 % de los fallos de forma correcta. Se diagnostica un fallo distinto al existente en un 5.55 % de los casos, y no se encuentra el fallo en un 13.88 % de los casos, teniendo un 0 % de falsos positivos, es decir, de indicar un fallo sin que exista. No obstante con un juego de pruebas más exhaustivo estos porcentajes pueden mejorar bastante. Sobre todo, hay que tener en cuenta, que entre los valores seleccionados para la evaluación, se encuentra un conjunto grande de los más desfavorables para acertar en el diagnóstico.

7.2.3. Aplicación de etiquetado múltiple

Fase *off-line*

La base de datos obtenida tras las simulaciones, tiene muchas *trayectorias* que son muy similares entre sí, a pesar de pertenecer a comportamientos distintos. Esto es un problema, porque como hemos observado en las pruebas previas, resultará muy difícil clasificar dichas trayectorias, y por tanto el diagnóstico obtenido al evaluar trayectorias similares puede resultar erróneo. La figura 7.6 muestra un ejemplo de dicha situación.

Para resolver el problema de la diagnosticabilidad de dichas trayectorias, se empleará la técnica del *etiquetado múltiple* (sección 6.7.1).

La magnitud seleccionada para medir el grado de similitud de dos trayectorias es la distancia euclídea. En concreto, para nuestro caso, se ha considerado que dos trayectorias no son diagnosticables, si la distancia euclídea entre ellas es menor al 10 % de la distancia euclídea que hay entre las dos trayectorias pertenecientes al comportamiento correcto que se encuentren más alejadas entre sí. En el caso de nuestro ejemplo, dicho valor es de 0.45. Así pues, las trayectorias de distinto comportamiento, cuya distancia euclídea sea menos de 0.45, se etiquetarán como pertenecientes a ambos comportamientos.

En esta ocasión, no vamos a realizar reducción del número de puntos de la trayectoria original.

Se calcularán los siguientes atributos para cada una de las trayectorias:

- Distancia al comportamiento perfecto (DP).
- Tiempo de Subida (t_r).
- Tiempo de Establecimiento (t_s).
- Sobreimpulso máximo (M_p).
- Tiempo de pico (t_p).

De esta forma, se generan las nuevas trayectorias con los nuevos atributos y etiquetas. Las trayectorias en la *base de datos tratada* tienen la siguiente estructura:

$$t_r, t_s, M_p, t_p, W_m[1], DP[1], ..., W_m[n], DP[n], ETIQUETA$$

Finalmente, se usa la herramienta de clasificación C4.5 para realizar el aprendizaje sobre la *base de datos tratada*. Para el presente caso, se ha obtenido un árbol con 27 nodos, proporcionando un porcentaje de error de 1.2 % sobre la base de datos de entrenamiento.

Resultados

Se han realizado un conjunto de simulaciones para test, a las que se ha aplicado el mismo tratamiento que a la base de datos de entrenamiento para poder ser así evaluadas con el árbol obtenido. Con objeto de comparar los resultados obtenidos con el etiquetado múltiple frente al etiquetado simple, se ha obtenido el árbol de decisión para los mismos datos de entrenamiento, pero sin realizar el etiquetado múltiple.

La tabla 7.3 muestra los resultados obtenidos para dichos tests:

Observamos que en los casos en los que los fallos vienen dados por valores de las constantes muy cercanas a la de los valores correctos, la diagnosis con etiquetado simple no es acertada. Esto se debe a que la estrecha relación existente entre las constantes de los componentes para este sistema, lo que provoca que alteraciones complementarias en estas constantes haga que el sistema evolucione de una misma forma. Sin embargo podemos comprobar que los resultados son notablemente mejores utilizando la ampliación del etiquetado múltiple. Entre las

Valor de la Constante			Diagnosis Correcta	Diagnosis con Etiquetado Simple	Diagnosis con Etiquetado Múltiple
Cm	Cc	Cs			
1	1	1.03	Cs Alta	Cs Alta	Cs Alta
1	1	1.07	Cs Alta	Cs Alta	Cs Alta
1	1	1.1	Cs Alta	Cs Alta	Cs Alta
1	1	1.5	Cs Alta	Cs Alta	Cs Alta
1	1	2	Cs Alta	Cs Alta	Cs Alta
1	1	3	Cs Alta	Cs Alta	Cs Alta
1	1.03	1	Cc Alta	Ok	Ok
1	1.07	1	Cc Alta	Cm Alta	Cc Alta Y Cm Alta
1	1.1	1	Cc Alta	Cm Alta	Cc Alta Y Cm Alta
1	1.5	1	Cc Alta	Cc Alta	Cc Alta
1	2	1	Cc Alta	Cc Alta	Cc Alta
1	3	1	Cc Alta	Cc Alta	Cc Alta
1.03	1	1	Cm Alta	Ok	Ok Y Cs Baja
1.07	1	1	Cm Alta	Cm Alta	Cc Alta Y Cm Alta
1.1	1	1	Cm Alta	Cm Alta	Cc Alta Y Cm Alta
1.5	1	1	Cm Alta	Cm Alta	Cm Alta
2	1	1	Cm Alta	Cm Alta	Cm Alta
3	1	1	Cm Alta	Cm Alta	Cm Alta
1	1	0.97	Cs Baja	Ok	Cs Baja Y Ok
1	1	0.93	Cs Baja	Cs Baja	Cs Baja
1	1	0.89	Cs Baja	Cs Baja	Cs Baja
1	1	0.85	Cs Baja	Cs Baja	Cs Baja
1	1	0.5	Cs Baja	Cs Baja	Cs Baja
1	1	0.1	Cs Baja	Cs Baja	Cs Baja
1	0.97	1	Cc Baja	Ok	Ok
1	0.93	1	Cc Baja	Cc Baja	Cc Baja Y Cm Baja
1	0.89	1	Cc Baja	Cc Baja	Cc Baja Y Cm Baja
1	0.85	1	Cc Baja	Cc Baja	Cc Baja Y Cm Baja
1	0.5	1	Cc Baja	Cc Baja	Cc Baja
1	0.1	1	Cc Baja	Cc Baja	Cc Baja

Valor de la Constante			**Diagnosis Correcta**	**Diagnosis con Etiquetado Simple**	**Diagnosis con Etiquetado Múltiple**
Cm	**Cc**	**Cs**			
0.97	1	1	Cm Baja	Ok	Ok
0.93	1	1	Cm Baja	Cc Baja	Cc Baja Y Cm Baja
0.89	1	1	Cm Baja	Cm Baja	Cc Baja Y Cm Baja
0.85	1	1	Cm Baja	Cm Baja	Cc Baja Y Cm Baja
0.5	1	1	Cm Baja	Cm Baja	Cm Baja
0.1	1	1	Cm Baja	Cm Baja	Cm Baja
0.99	0.98	1.02	Ok	Ok	Ok
1	1.02	1.02	Ok	Ok	Ok
0.98	1	0.98	Ok	Ok	Ok
0.98	1.02	1.02	Ok	Ok	Ok
0.99	1.01	1.01	Ok	Ok	Ok
1.01	1	0.99	Ok	Ok	Ok

Tabla 7.3: Resultados obtenidos

opciones ofrecidas en el caso de la diagnosis con etiquetado múltiple, la diagnosis correcta se encuentra entre las ofrecidas en la mayoría de los casos.

Es importante señalar que, in los casos de test donde le comportamiento se aleja bastante del comportamiento correcto, la diagnosis ofrecida es única y correcta. En el conjunto de tests presentados, la diagnosis correcta se obtiene en un 58.33 % de las ocasiones. Se ofrece un diagnóstico incorrecto en el 2.7 % de los casos y el fallo no se detecta en 8.33 %. Por otro lado, hay que resaltar que no se produce ningún caso de falso positivo, es decir, nunca se diagnostica un fallo si no existe.

Los porcentajes presentados para el estudio presentan valores aceptables, teniendo en cuenta que los casos de test no siguen una distribución uniforme de los valores de fallos, sino que se ha seleccionado un conjunto de valores de fallos donde abundan los casos más desfavorables, que son aquellos muy cercanos a los valores correctos.

7.3. Modelo detallado de lazo de control para un motor DC

7.3.1. Descripción del sistema

Una vez probada la validez de la metodología con un modelo simplificado del sistema a diagnosticar, el siguiente paso será modelar el sistema de forma que represente los componentes involucrados y los posibles fallos de dichos componentes.

El modelo seleccionado está compuesto por un motor de corriente continua de excitación independiente, que es alimentado por un rectificador de tres fases. El motor es alimentado por el circuito rectificador de tres fases, a través de un 'chopper', compuesto por un transistor IGBT, y un diodo de libre circulación. La fuerza del motor es controlada por la intensidad que circula por la armadura que es regulada por un lazo de control de corriente. La velocidad de giro del motor se controla mediante un controlador PI externo, que proporciona la referencia de intensidad para el lazo de control de corriente (fig. 7.8).

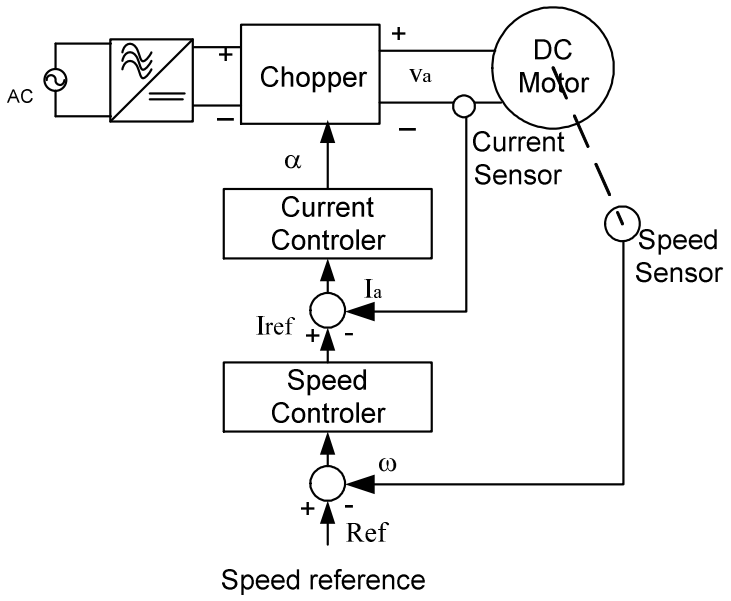

Figura 7.7: Lazo de control

El motor mueve una polea, que es modelada mediante la inercia J, el coeficiente de fricción B y el par de la carga T_l. Este motor realizará un trabajo repetitivo, acelerando desde la posición de reposo hasta alcanzar la velocidad angular de refe-

rencia, siempre con la misma carga. Esta podría ser la situación de un motor que mueve una bomba centrifugadora. Este sistema ha sido modelado con Simulink® usando componentes de la Toolbox SimPowerSystems® (Figuras 7.8 y 7.9).

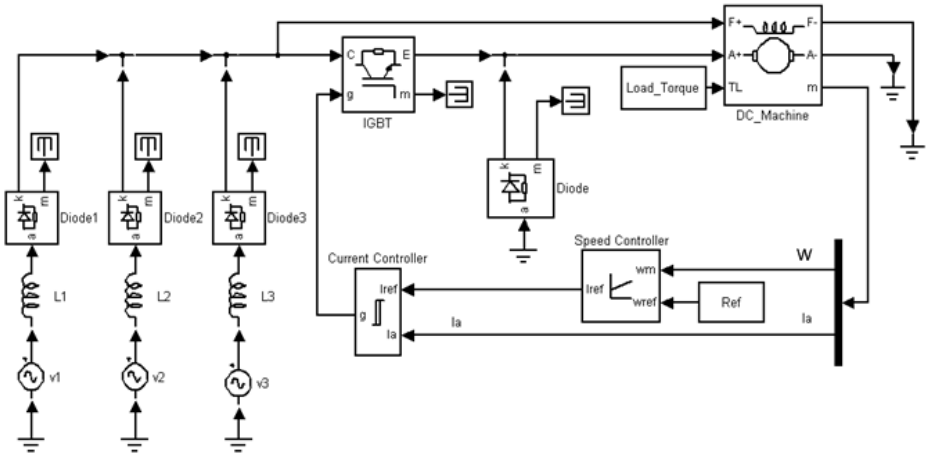

Figura 7.8: Modelo en Simulink® del sistema

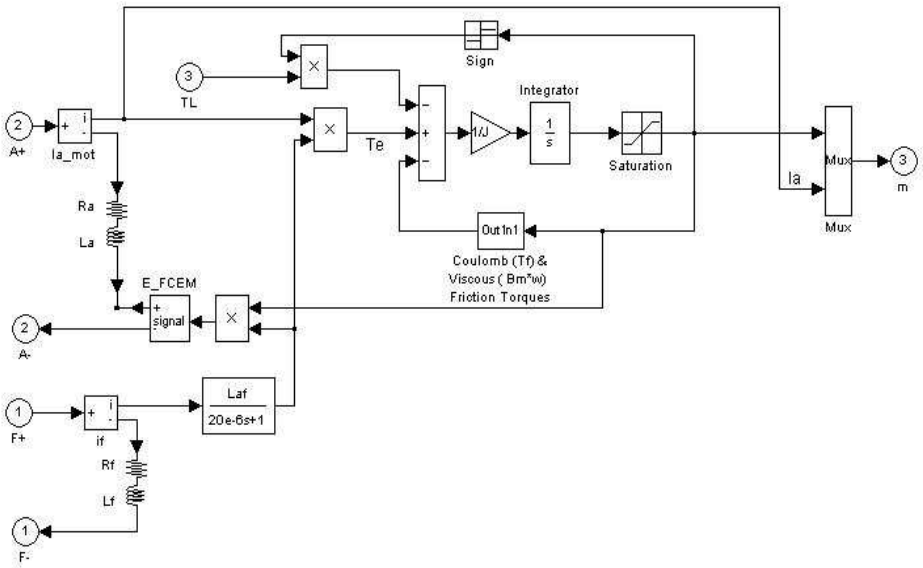

Figura 7.9: Modelo en Simulink® del motor DC

El motor se modela mediante su circuito equivalente (fig. 7.10), consistente en una bobina L_a y una resistencia R_a en serie con la fuerza electromotriz (EFM) E.

La reacción devuelta EFM es proporcional a al velocidad del motor (7.9).

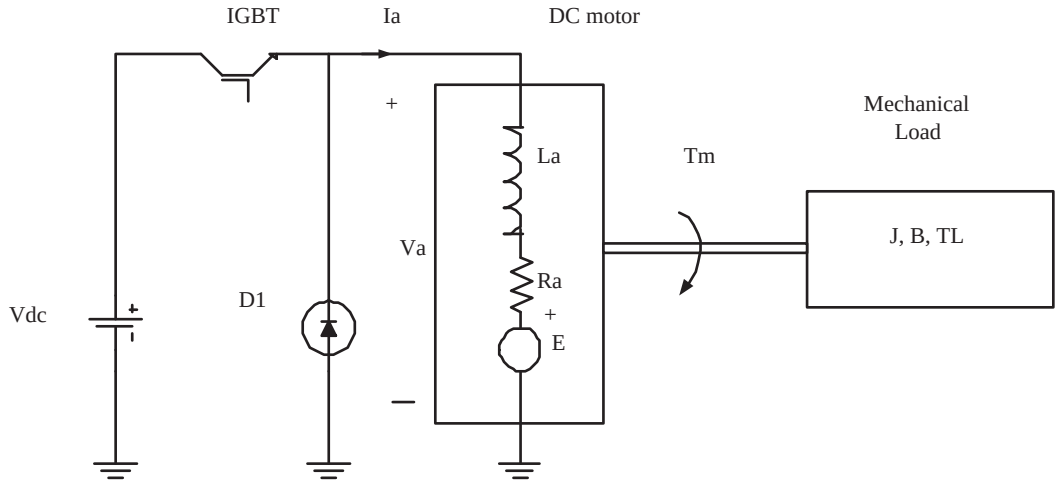

Figura 7.10: Circuito equivalente del motor de continua

$$E = K_E \omega \tag{7.9}$$

donde K_E es la constante de tensión del motor y ω es la velocidad del motor.

La constante de tensión del motor K_E es proporcional a la corriente del campo I_f (7.10).

$$K_E = L_{af} I_f \tag{7.10}$$

donde L_{af} es la inductancia entre el campo y la armadura.

El par desarrollado por el motor es proporcional a la corriente de armadura I_a (7.11).

$$T_m = K_T I_a \tag{7.11}$$

donde K_T es la constante de par del motor.

La constante de par del motor es igual a la constante de tensión (7.12).

$$K_T = K_E \tag{7.12}$$

La fuente de continua (V_{dc}) es la salida del circuito rectificador de tres fases. La tensión media de salida es obtenida por (7.13).

$$V_{AV} = \frac{3}{\pi} V_M \sin \frac{\pi}{3} \tag{7.13}$$

Nombre	**Amplitud (V)**	**Fase (deg)**	**Frecuencia (Hz)**
v1	$220\sqrt{2}$	0	50
v2	$220\sqrt{2}$	-120	50
v3	$220\sqrt{2}$	120	50

Tabla 7.4: Parámetros de la fuente de tensión alterna

Los parámetros de la tensión alterna se muestran en la tabla 7.4.

El transitor IGBT es disparado por una señal de pulso modelado en anchura (PWM) para controlar la salida media.

Las formas de onda teóricas que ilustran la operación de 'chopper' se muestran en la figura (7.11).

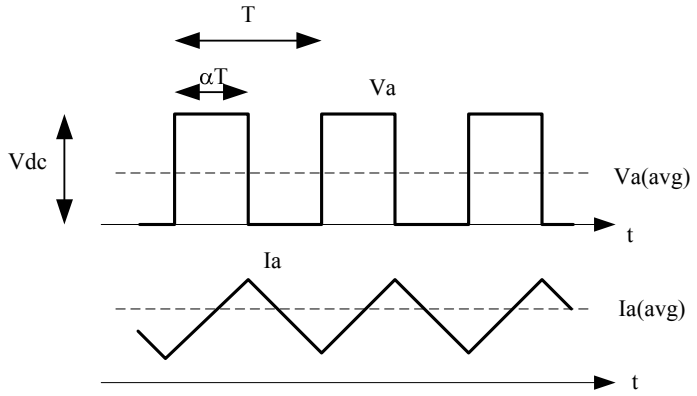

Figura 7.11: Señales para la operación de *Chopper*

La tensión media de la armadura es función directa del ciclo de 'chopper' impuesto α (7.14).

$$V_a(avg) = \alpha V_{dc} \tag{7.14}$$

Esta relación es válida, sólo cuando la corriente de la armadura es continua. En el estado estacionario, la corriente media en la armadura es igual a (7.15).

$$I_a(avg) = \frac{V_a(avg) - E}{R_a} \tag{7.15}$$

El rizo de corriente pico a pico es (7.16).

$$\Delta_i = \frac{V_{dc}}{R_a} \frac{(1 - e^{\alpha r} + e^{-r} - e^{-(1-\alpha)r})}{1 - e^{-r}} \tag{7.16}$$

donde α es el ciclo impuesto y r es la relación entre el periodo de 'chopper' y la constante de tiempo eléctrica del motor DC (7.17).

$$r = \frac{T}{L_a/R_a} \tag{7.17}$$

El transitorio seleccionado para las simulaciones comienza en la posición del motor parado, y acelera hasta alcanzar una velocidad de 50 rad/sec. Los parámetros del sistema seleccionados para las simulaciones son los mostrados en la tabla 7.5.

Parámetro	**Valor**	**Unidad**
Potencia Nominal	3.9	KW
Resistencia de Armadura	0.6	Ω
Inductancia de Armadura	0.0012	H
Resistencia de Campo	240	Ω
Inductancia de Campo	120	H
Autoinducción mutua	1.8	H
Par de Fricción	0.5	N.m
Par de Carga torque	10	N.m
Inercia	1	$Kg.m^2$
Coeficiente de fricción	0.05	N.m.s
Ganancia proporcional del controlador PI	3	Kp
Ganancia Integral del controlador PI	0.5	Ki
Limitación de corriente del controlador PI	200	A
Banda de Histéresis del controlador de corriente	3	A

Tabla 7.5: Parámetros de la simulación

Se va a considerar un intervalo de tolerancia en los valores nominales de cada componente. Estos componentes se considerará que están libres de fallos siempre y cuando sus valores estén dentro de dicho intervalo. Con esto se modelan las pequeñas alteraciones que puedan producirse en un sistema sin que se considere que éste está fallando. Dichas desviaciones pueden venir dadas por alteraciones externas, como pueden ser, por ejemplo, las condiciones meteorológicas. El rango de tolerancia considerado se ha establecido en un 4 % de los valores nominales de cada componente.

7.3.2. Identificación de los fallos

Se consideran dos tipos de fallos a diagnosticar:

Fallos de Ruptura. Este tipo de fallos ocurren cuando el componente que falla se rompe, y por tanto deja de funcionar totalmente. Para el sistema que nos ocupa, se considerarán los siguientes fallos de ruptura:

- Perdida de una fase. La fuente de alimentación trifásica, pierde una de sus fases.
- Perdida de dos fases. La fuente de alimentación trifásica, pierde dos de sus fases.
- Cortocircuito del transistor IGBT . El transistor IGBT se cortocircuita, permitiendo el paso de corriente de forma continuada.
- Cortocircuito del diodo de libre circulación. El diodo de libre circulación se cortocircuita, permitiendo pasar la corriente a través de él de forma continuada.
- Apertura del diodo de libre circulación. El diodo de libre circulación se queda abierto, no dejando pasar nunca la corriente a través de él.

Fallos de Desgaste. Este tipo de fallos ocurren cuando los componentes van perdiendo sus propiedades, debidos al desgaste, lo que repercute en una alteración perceptible en la forma de actuar del sistema. Serán considerados los siguientes fallos de desgaste para el modelo que nos ocupa:

- Se cortocircuitan algunas espiras en la bobina de Armadura. Esto produce un decremento de la inductancia y resistencia de la bobina de armadura. La relación entre el número de espiras cortocircuitadas y el decremento de inductancia y resistencia producido viene dado por las ecuaciones 7.18 y 7.19.

$$\Omega = K\frac{L}{S} \tag{7.18}$$

$$L = \frac{\mu_r \mu_0 N^2 A}{L_c} \tag{7.19}$$

- Se cortocircuitan algunas espiras en la bobina de Campo. El fallo es igual al anterior, pero en esta ocasión producido en la bobina de campo.
- Incremento del rozamiento del eje. Se produce un exceso de rozamiento en el eje de giro, debido al desgaste, el exceso de temperatura o la falta de lubricación. Esto es modelado como un incremento del torque de fricción.

Cada fallo de ruptura se ha modelado realizando los cambios en el modelo para reflejar el fallo. Por ejemplo, la caída de una de las fases se representa en el modelo de la figura 7.12.

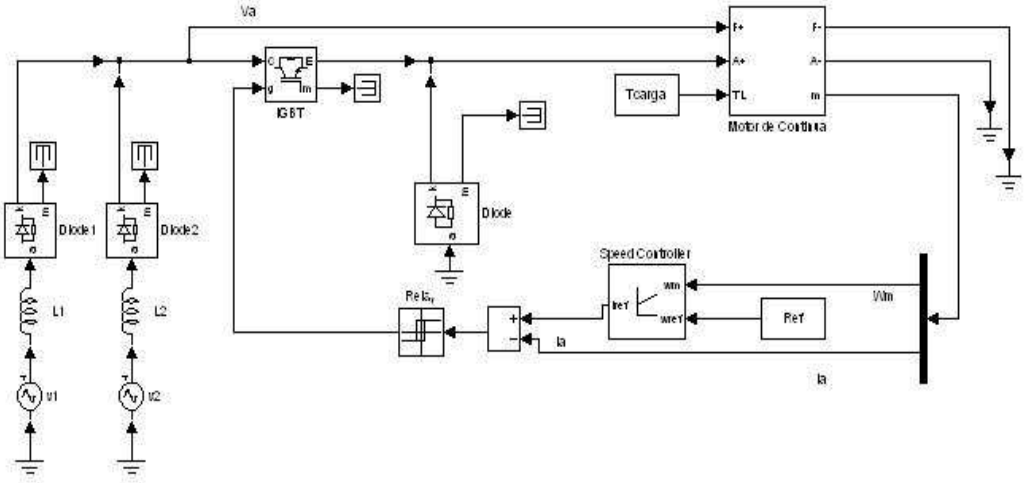

Figura 7.12: Modelo Simulink® del sistema con el fallo de 1 fase

Para los fallos de desgaste, se modifica el valor nominal del componente que falla. Para los casos de cortocircuitos de las espiras de la bobina, se considera un reducción de hasta un 40 % en la resistencia e inductancia de dichas bobinas, dependiendo de la cantidad de espiras cortocircuitadas. Para el fallo de rozamiento del eje, se considera un aumento en el torque de fricción de hasta 10 N.m.

Los datos recogidos del sistema, se corresponden con la velocidad de giro del rotor (ω) y la señal de control I_{ref}. En la Figura (Fig. 7.8) se puede apreciar la representación gráfica de dichas señales.

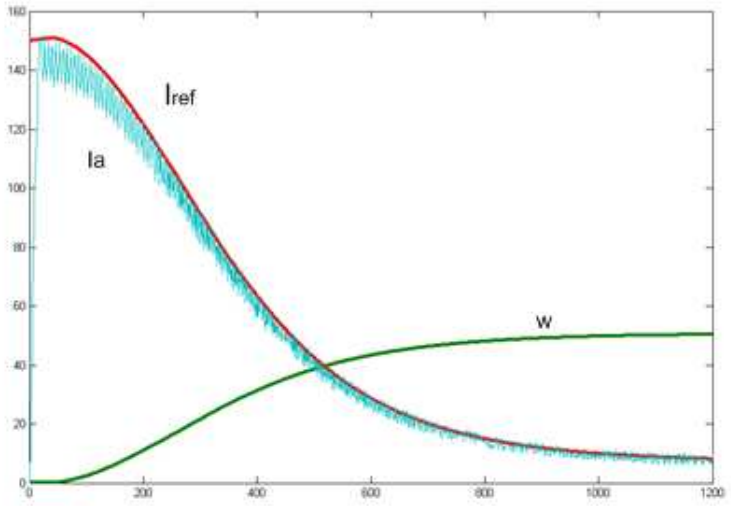

Figura 7.13: Evolución de las variables observables del sistema

7.3.3. Clasificación Múltiple

Se han realizado treinta simulaciones de cada comportamiento para el modelo descrito. Los valores nominales de los componentes se han seleccionado siguiendo una distribución aleatoria dentro del intervalo adecuado al comportamiento simulado. Los datos recogidos se corresponden con la velocidad de giro del rotor (W_m) y la señal de control (Cv)

A la *base de datos etiquetada* se ha aplicado el tratamiento de segmentación y normalización para reducir el número de elementos que compone cada *trayectoria*. Una vez realizada la reducción se ha calculado la *inclinación relativa* de cada segmento para obtener finalmente la *base de datos tratada*.

Para obtener el árbol de decisión se ha usado la herramienta de clasificación C5.0®. El árbol obtenido tiene un tamaño de 9 nodos.

Para resolver el problema de la diagnosticabilidad, en este modelo se ha usado también la técnica del etiquetado múltiple. Así pues, se ha replicado la misma base de datos tratada, para obtener una base de datos por cada comportamiento, donde las únicas etiquetas serán la pertenencia de la trayectoria al comportamiento o la no pertenencia. Cada una de dichas bases de datos ha dado lugar a un árbol de decisión de cada comportamiento. Los tamaños de dichos árboles van desde 1 a 6 nodos.

Resultados

Para evaluar los árboles de decisión obtenidos, se han generado 10 nuevas simulaciones de cada uno de los comportamientos. Cada nueva simulación ha sido tratada en la misma forma en que lo fueron los datos de entrenamiento para poder ser evaluadas por los árboles de decisión obtenidos.

Los resultados obtenidos se ofrecen en la Tabla 7.6.

Fallo	**Comportamiento**	**Diagnóstico con Árbol Simple**	**Diagnostico con Múltiples Árboles**
OK	Ausencia de Fallo	100 % OK	80 % OK 20 % OK or AW
AW	Cortocircuito en la Bobina de Armadura	80 % AW 20 % OK	60 % AW 20 % AW or OK 20 % AW or BF
FW	Cortocircuito en la Bobina de Campo	90 % FW 10 % BF	80 % FW 10 % FW or F1 10 % ??
BF	Exceso de Rozamiento	90 % BF 10 % FW	80 % BF 20 % BF or FW
F1	Falla 1 Fase	100 % F1	90 % F1 10 % ??
F2	Fallan 2 Fases	100 % DS	100 % F2 or DS
IG	Cortocircuito del IGBT	100 % IG	100 % IG
DS	Cortocircuito del Diodo	100 % DS	100 % DS or F2
DO	Diodo Abierto	100 % DO	100 % DO

Tabla 7.6: Resultados

Es importante hacer notar que los fallos $F2$ y DS no son discriminables, porque ambos fallos producen exactamente la misma medida de los sensores.

Como puede verse en la tabla de resultados, los fallos de ruptura son perfectamente diagnosticados con la clasificación de un sólo árbol, salvo para los fallos $F2$ y DS, que ya hemos comentado que no son discriminables. En este caso, la diagnosis del fallo $F2$ es siempre errónea, indicando como diagnostico la presencia

del fallo *DS*. Esta situación es diferente en la evaluación de los árboles individuales, puesto que dicha evaluación ofrece como diagnóstico para ambos fallos, las dos posibilidades, sin ser capaz de discernir de cual se trata. Esta situación es la más próxima a la realidad, puesto que con las lecturas disponibles, es imposible saber cual de ellos es el fallo correcto.

Los fallos de desgaste son diagnosticados de forma más imprecisa en ambos casos. Esto se debe a que cuando los valores de los componentes que fallan se encuentran muy próximos a los valores correctos, la lectura de los sensores es muy similar para distintas situaciones. El uso de la clasificación con árboles múltiples, hace que dichas situaciones se presenten de forma ambigua, ofreciendo varios posibles diagnósticos.

7.3.4. Clasificación con Boosting

Con objeto de mejorar la precisión de los resultados obtenidos en la clasificación anterior, se pretende realizar una clasificación mediante la técnica de *boosting* con los datos de la base de datos tratada presentada en la sección anterior (sección 7.3.3). Para tal fin, se utiliza la herramienta de clasificación C5.0, que dispone de la posibilidad de seleccionar el número de clasificadores a emplear para el boosting, como puede apreciarse en la figura (7.14. Se han seleccionado 3 clasificadores. El número impar de la selección provoca que no pueda haber empate en la votación a la hora de evaluar las nuevas observaciones.

Para comparar los resultados, se ha generado un único árbol de decisión (sin usar boosting) que tiene un tamaño de 10 nodos. Los árboles generados con la opción de boosting, tienen tamaños de 13, 11 y 9 nodos respectivamente.

Resultados

Para evaluar los árboles de decisión obtenidos, se han generado 10 nuevas simulaciones de cada uno de los comportamientos. Cada nueva simulación ha sido tratada en la misma forma en que lo fueron los datos de entrenamiento para poder ser evaluadas por los árboles de decisión obtenidos.

Para obtener un diagnóstico único en la evaluación de la nueva observación con los tres árboles obtenidos mediante boosting, se ha realizado una votación sin

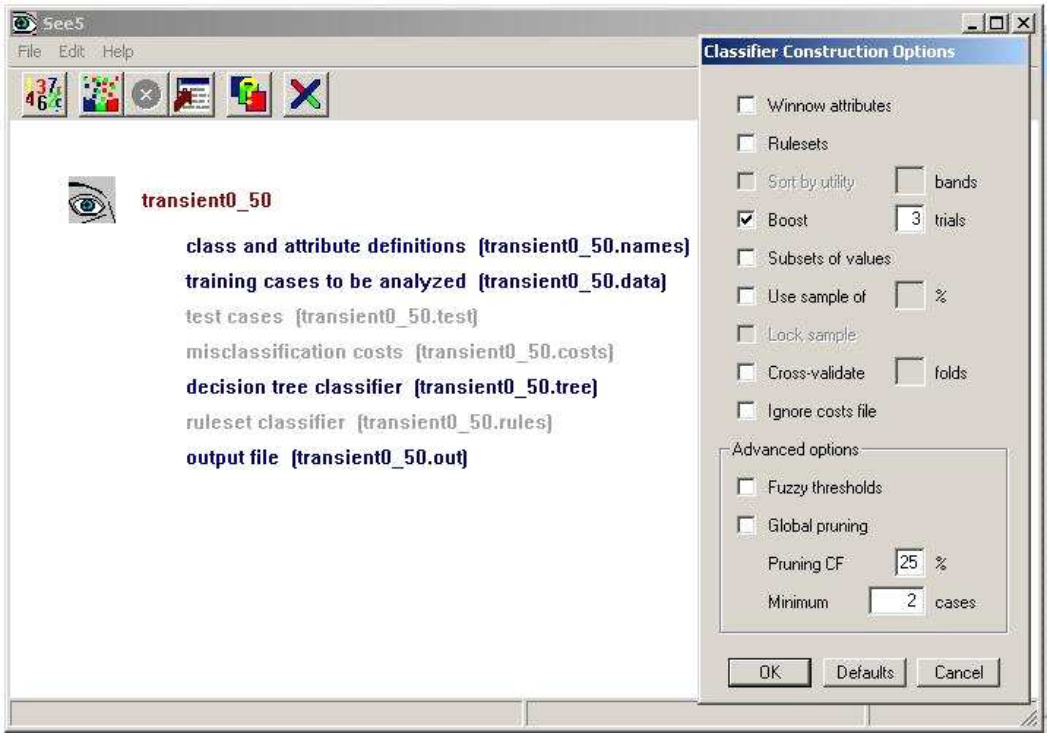

Figura 7.14: Herramienta C5.0®

ponderación entre los resultados obtenidos. Los resultados obtenidos con ambos métodos de clasificación son los mostrados en la tabla 7.7.

Como se puede observar en los resultados, lo primero que hay que destacar, es que nunca se ofrece un falso positivo, es decir, nunca se diagnostica un fallo si éste no existe.

Recordemos que los fallos F2 y DS no son diagnosticables, y al no tratar el problema de la diagnosticabilidad en este caso, produce que el fallo F2 sea siempre diagnosticado erróneamente. Salvo el citado caso de F2, el resto de los fallos de ruptura son siempre diagnosticados de forma correcta con ambas opciones.

Con respecto a los fallos de desgaste, a pesar de que la diagnosis es más imprecisa, debido a la cercanía en ocasiones del comportamiento correcto, vemos como la clasificación mediante boosting es capaz de mejorar los resultados del diagnostico mediante el árbol único. Concretamente, en el caso de los fallos AW y BF, la mejora producida por la clasificación mediante boosting es de un 10 %.

Fallo	**Comportamiento**	**Diagnosis con Árbol Simple**	**Diagnosis con Boosting**
(OK)	Ausencia de Fallo	100 % OK	100 % OK
(AW)	Cortocircuito en la Bobina de Armadura	80 % AW 20 % OK	90 % AW 10 % OK
(FW)	Cortocircuito en la Bobina de Campo	90 % FW 10 % BF	90 % FW 10 % BF
(BF)	Exceso de Rozamiento	90 % BF 10 % FW	100 % BF
(F1)	Falla 1 Fase	100 % F1	100 % F1
(F2)	Fallan 2 Fases	100 % DS	100 % DS
(IG)	Cortocircuito del IGBT	100 % IG	100 % IG
(DS)	Cortocircuito del Diodo	100 % DS	100 % DS
(DO)	Diodo Abierto	100 % DO	100 % DO

Tabla 7.7: Resultados

7.4. Motor DC de 0,61 Kw controlado por 'chopper'

7.4.1. Descripción del sistema

Tras los resultados obtenidos con los modelos, la metodología ha sido probada a un motor de corriente continua de excitación independiente (ALECOP AL 506; 220V/0.61 KW), que es alimentado por un rectificador AC-DC.

El campo del motor es alimentado a través de rectificador AC-DC comercial (ALECOP GTR200) que permite seleccionar manualmente el voltaje de salida.

La armadura se alimenta a través de otro rectificador AC-DC. El voltaje de salida del rectificador es modificado mediante un proceso de "*chopper*" llevado a cabo por un transistor IGBT y un diodo de libre circulación. El transistor IGBT se excita mediante una señal PWM, cuyo periodo es controlado por un PID que lo ajusta dependiendo de la velocidad seleccionada y la medida. El controlador PID es implementado por software en el PC que recoge los datos proporcionados por

el sensor. El sensor que mide la velocidad del motor es un tacómetro, cuyo señal es acondicionada antes de poder ser leída.

En el Apéndice A se describe el hardware desarrollado para el acondicionamiento de la señal, y para el 'chopper' de la corriente de alimentación del rotor.

Este motor mueve una carga fija, realizando una tarea recurrente entre el estado de reposo y la referencia proporcionada. Este podría ser el escenario de un motor que maneja una bomba centrifugadora.

La figura 7.15 muestra el diagrama de bloques del sistema, mientras que la figura 7.16 muestra el sistema propiamente dicho.

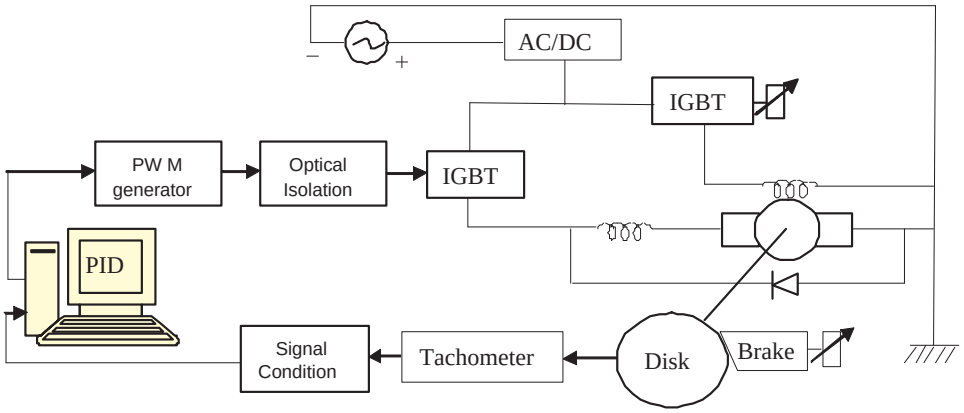

Figura 7.15: Diagrama de bloques del sistema

El transitorio considerado para los ensayos, va desde la posición de reposo del sistema hasta alcanzar una referencia de 700 *rpm*. El voltaje que se considera correcto para alimentar el campo oscila entre los 115 y los 125 voltios. Se considera también que el comportamiento correcto del sistema evoluciona sin ninguna fuerza de frenado sobre el disco.

7.4.2. Identificación de los fallos

Los fallos que pretendemos diagnosticar son de dos tipos distintos:

Fallos abruptos. La ocurrencia de estos fallos supone la ruptura total del componente defectuoso, lo que implica que deje de funcionar totalmente. Consideraremos los siguientes fallos abruptos:

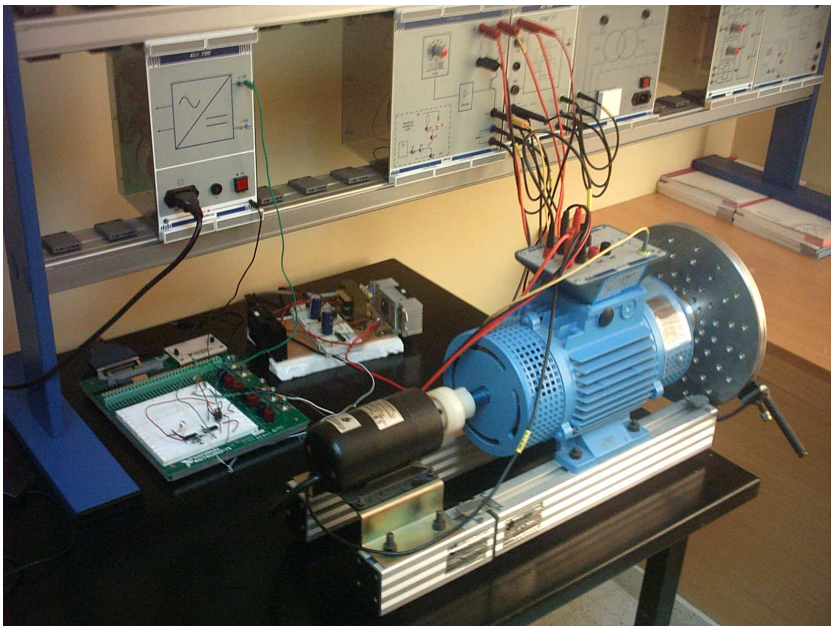

Figura 7.16: Sistema real

- Se cortocircuita la bobina de compensación. Esto produce que el motor alcance su velocidad de referencia de forma diferente a como lo haría con la bobina intacta. Para reproducir este fallos se cortocircuita la bobina de compensación mediante un cable. Este fallo será etiquetado como *F1*.
- Se pierde la conexión de tierra. Este fallo ocurre cuando se desconecta o se rompe la conexión de la toma de tierra. Para reproducir este fallo basta con desconectar el cable de tierra. Este fallo es etiquetado como *F2*.
- Ruptura del transistor IGBT. El transistor IGBT deja de conducir. Este fallo se reproduce desconectando la alimentación del IGBT. El fallo se etiqueta como *F3*.
- Cortocircuito del transistor IGBT. El transistor IGBT siempre deja pasar la corriente. Para reproducir este fallo se hace un bypass al transistor IGBT. Este fallo se etiqueta como *F4*.

Fallos incipientes. Este tipo de fallos ocurren cuando algún componente va perdiendo gradualmente sus propiedades. Esto produce una alteración en el comportamiento del sistema que será más perceptible mientras más se aleje

el componente de sus valores correctos. El componente sigue funcionando, pero con sus propiedades alteradas. Se considerarán los siguientes fallos incipientes:

- El voltaje que alimenta el campo es inferior al adecuado. Para reproducir este fallo basta con modificar manualmente el voltaje de salida de rectificador AC/DC(ALECOP GTR200), seleccionando valores inferiores al considerado correcto. Este fallo será etiquetado como *F5*.
- El voltaje que alimenta el campo es superior al adecuado. Para reproducir este fallo se procede de forma contraria al especificado para el error previo. Este fallo se etiqueta como *F6*.
- Incremento de rozamiento en el rodamiento del eje. Debido al desgaste, el rodamiento del eje gira con un rozamiento superior al considerado correcto. Para reproducir este fallo es necesario operar sobre el freno que el motor tiene conectado al disco que hace girar. Este fallo se etiqueta como *F7*.

Cada fallo abrupto se ha conseguido manipulando el sistema, tal y como se indica, para cada uno de los fallos.

Para los fallos incipientes, los valores de los componentes involucrados se han modificado gradualmente en cada ensayo, desde valores muy cercanos a los correctos hasta otros más lejanos. Estos son los casos del incremento y decremento del voltaje que alimenta el campo. Para estos fallos, el valor más bajo considerado es de 75 voltios, y el mayor de 165 voltios.

El incremento de fricción se consigue operando sobre el freno de forma gradual.

El transitorio entrenado va desde la posición de parada hasta alcanzar una referencia de 700 rpm. Se considera que los voltajes correctos para la alimentación del estator están en el intervalo entre 115 voltios y 125 voltios. Se considera que cuando el sistema está funcionando de forma correcta, el eje está libre del rozamiento del freno.

7.4.3. Clasificación con Boosting

Con objeto de obtener la *base de datos etiquetada*, se han recogido un serie de datos de funcionamiento con el sistema descrito anteriormente. Concretamente,

se han realizado 50 ensayos por cada uno de los comportamientos de fallo y 100 para el caso de la ausencia de fallo. Los valores de voltaje que alimenta el estator en el comportamiento sin fallos se han seleccionado siguiendo una distribución gaussiana dentro del rango de valores correctos.

Una vez creada la base de datos de trayectorias etiquetadas, se ha procedido a la segmentación y normalización de las trayectorias. Para cada trayectoria se ha calculado, además, la *inclinación relativa* de cada segmento, y se ha añadido el número de *cambios de tendencia* de la trayectoria. Finalmente, el resultado será la *base de datos tratada* que nos servirá para el aprendizaje.

Se ha desarrollado una aplicación (Figura 7.20) que automatiza todo el proceso de tratamiento de la base de datos etiquetada, para conseguir la base de datos tratada.

La base de datos tratada tiene el 43 % del tamaño que tenía la base de datos etiquetada, con lo cual, a pesar de haber incluido información adicional en el proceso de tratamiento, la reducción de tamaño conseguido con la segmentación y la posterior normalización es bastante significativa.

Finalmente, la herramienta de clasificación C5.0® ha sido empleada con la base de datos tratada. Se han seleccionado 5 ensayos de boosting, para obtener 5 árboles de decisión distintos. Los tamaños de los árboles de decisión son respectivamente de 10, 10, 10, 11 y 13 nodos.

Para validar la precisión de los árboles obtenidos, se ha usado validación cruzada con 10 subconjuntos distintos. La Tabla 7.8 muestra los resultados obtenidos.

(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	<-classified as
98	1					1		(a): class OK
	50							(b): class F1
		49				1		(c): class F2
			50					(d): class F3
				50				(e): class F4
2					48			(f): class F5
2						46	2	(g): class F6
		2				2	46	(h): class F7

Tabla 7.8: Resultados de la Validación Cruzada

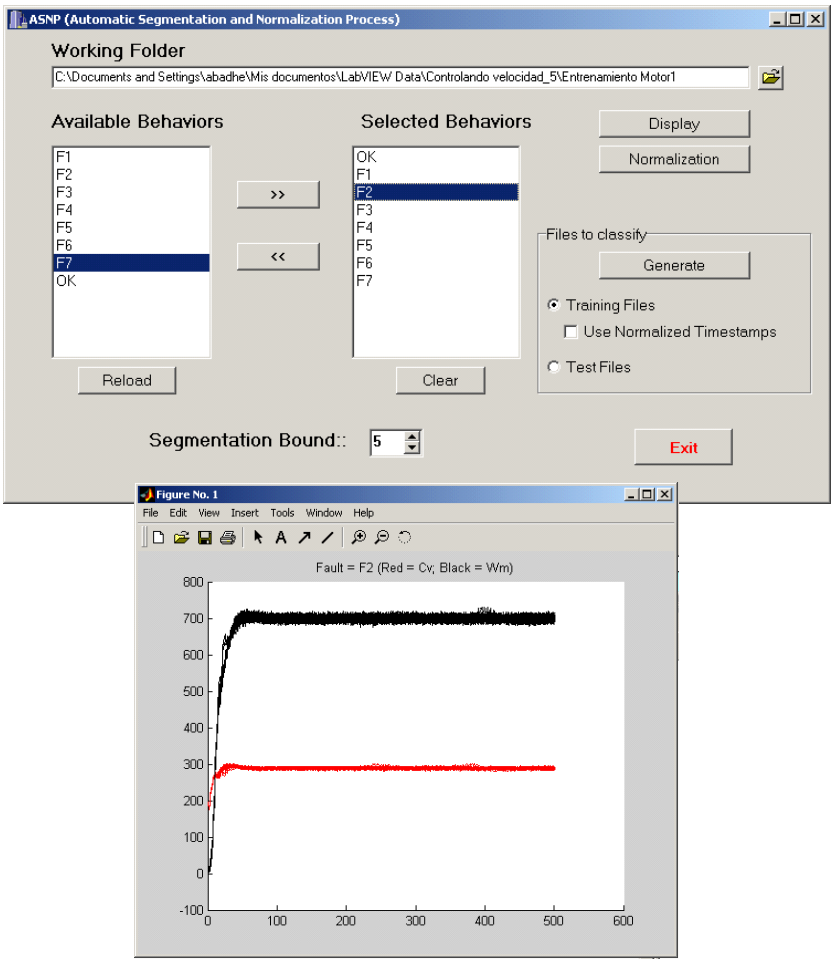

Figura 7.17: Herramienta que automatiza el tratamiento

Resultados

Para evaluar los árboles de decisión obtenidos, se ha procedido a la ejecución de 20 nuevos ensayos de cada comportamiento. Los resultados obtenidos se muestran en la tabla 7.12.

(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	<-classified as
19						1		(a): class OK
	20							(b): class F1
		15				2	3	(c): class F2
			20					(d): class F3
				20				(e): class F4
1					19			(f): class F5
						8	12	(g): class F6
						1	19	(h): class F7

Tabla 7.9: Casos de Test del motor 1

Puesto que el objetivo de la metodología de diagnosis presentada es conseguir que sea compatible con un producto que se produzca en serie, se han evaluado los árboles de decisión obtenidos con los datos de una unidad diferente del mismo motor. Para este nuevo motor no se ha realizado ningún entrenamiento previo, tan sólo se recogen 20 nuevos datos de ensayo para el mismo transitorio ensayado con el otro motor. Los nuevos ensayos del *motor2* son ahora evaluados con los árboles obtenidos tras el entrenamiento del *motor1*. Los resultados obtenidos se muestran en la Tabla 7.13.

(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	<-classified as
17						2	1	(a): class OK
10	10							(b): class F1
		13				1	6	(c): class F2
			20					(d): class F3
				20				(e): class F4
					20			(f): class F5
						19	1	(g): class F6
							20	(h): class F7

Tabla 7.10: Casos de Test del motor 2

Como puede verse en la tabla de resultados, la precisión obtenida en el diagnóstico es bastante buena en la mayoría de los casos. El comportamiento correcto es diagnosticado correctamente en un 95 % de los casos con el *motor1*, y en un

85 % de los casos con el *motor2*. Los errores producidos en la diagnosis del comportamiento correcto se producen con los fallos *F6* y *F7*. Estos fallos son fallos incipientes, y para dichos fallos, la evolución de las trayectorias es muy similar a las del comportamiento correcto cuando dichos fallos se producen por desviaciones pequeñas del comportamiento correcto. Algo similar ocurre con los fallos *F6* y *F7* entre sí. Ambos fallos, producen una evolución muy similar de las trayectorias, y ciertas desviaciones en sus valores producen una evolución muy parecida del sistema. Por otra parte, resulta curioso observar como la confusión entre los fallos *F6* y *F7* es más habitual en el *motor1* que en el *motor2*, a pesar de que los datos de entrenamiento son del primero.

Los peores resultados se obtienen para la diagnosis del fallo *F1* sobre el *motor2*, pero recordemos que sobre este motor no se ha realizado entrenamiento alguno. Con respecto a los errores producidos en la diagnosis del fallo *F2*, este fallo es fácilmente discriminable por el número de *cambios de tendencia*, pero la herramienta de clasificación selecciona un valor de separación demasiado cercano a uno de los comportamientos, y esta es la causa de las confusiones. Esta situación sería fácilmente corregible si el valor de la regla se estableciese algo más lejos de los valores de uno de los comportamientos.

Para demostrar la rapidez de la evaluación de los árboles, y que la metodología puede ser aplicada *on-line*, se ha desarrollado una aplicación que implementa dichos árboles y evalúa la nueva observación *on-line*. Los resultados obtenidos han sido satisfactorios, tanto en precisión como en velocidad. La Figuras 7.18 y 7.19 muestran dos situaciones diagnosticadas con la aplicación desarrollada. La aplicación ha sido implementada en LabView®.

7.4.4. Aplicación de DISETRA

Como se ha comentado en la sección anterior, existen una serie de fallos de diagnóstico del fallo F2 a pesar de que dicho fallo es discriminable perfectamente por el número de cambios de tendencia. Esto se produce porque la regla establece el valor de corte muy cerca de los valores de uno de los comportamientos.

El algoritmo *DISETRA* presentado en la sección 6.8 busca, precisamente, el valor de separación más alejado entre las trayectorias de distintos comportamientos.

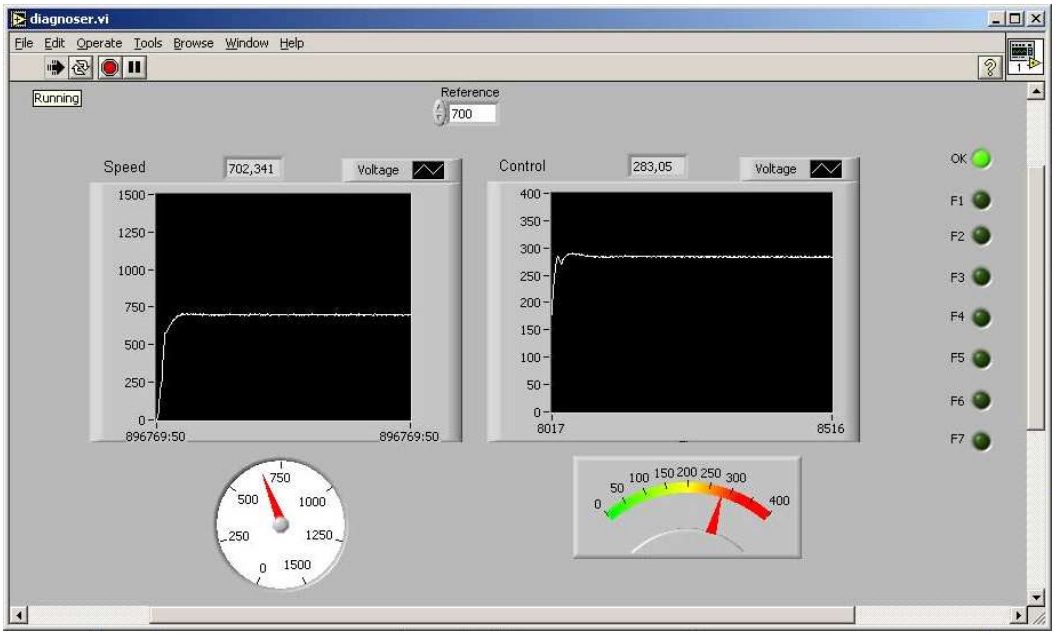

Figura 7.18: Diagnosis *on-line* del comportamiento correcto

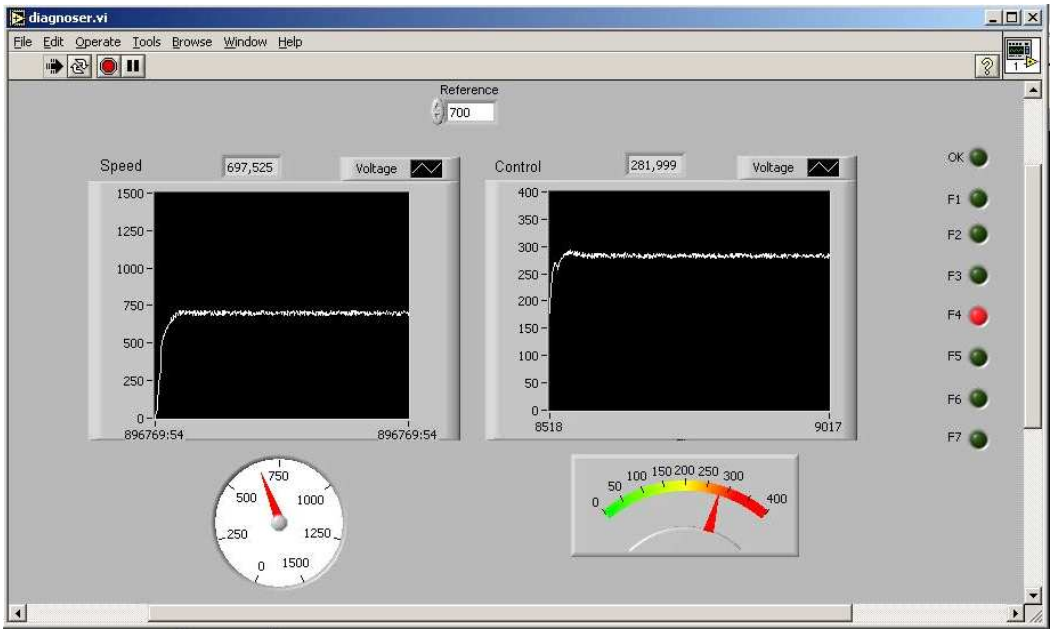

Figura 7.19: Diagnosis *on-line* del fallo F4

Pars evaluar el algoritmo *DISETRA*, se han realizado 50 nuevos ensayos de cada uno de los comportamientos a diagnosticar. Una vez etiquetados dichos ensayos obtendremos la *base de datos etiquetada*.

A la *base de datos etiquetada* se le aplica el algoritmo DISETRA. El valor del umbral del error seleccionado para la segmentación se ha establecido en 20, y para seleccionar el instante de tiempo en el que generar las reglas se ha optado por la opción de la máxima separación.

Dicho algoritmo ha establecido tres comportamientos *totalmente separables F2*, *F3* y *F4*, cuatro comportamientos *separables individualmente F1*, *F5*, *F6* y *F7* y uno *no separable OK*.

La tabla 7.11 muestra el conjunto de reglas obtenidas con DISETRA.

```
if WM(338) > 970.4
    disp('F3');
elseif WM(69) < 344.2
    disp('F4');
elseif SEGSWM > 196
    disp('F2');
elseif (WM(32)<655.9 & WM(9)>190.3 & CV(120) <282.5)
    disp('F1');
elseif (WM(32)>655.9 & WM(33)>660.6 & CV(474)<284.5)
    disp('F5');
elseif (CV(120)>282.5 & WM(33)<660.6 & WM(7)>96.6)
    disp('F6');
elseif (WM(9)<190.3 & CV(474)>284.5 & WM(7)<96.6)
    disp('F7');
else
    disp('OK');
end
```

Tabla 7.11: Conjunto de reglas obtenidos con DISETRA

Para automatizar el proceso se ha desarrollado la aplicación mostrada en la figura 7.20.

Resultados

Para probar la eficiencia del conjunto de reglas obtenidos se han realizado un conjunto de 20 nuevas pruebas de cada comportamiento. El resultado de la evaluación de dichas pruebas es el mostrado en la tabla 7.12.

Puesto que la metodología pretende poder ser extendida al diagnóstico de una serie de productos fabricados en serie, se han realizado 20 nuevas pruebas con un

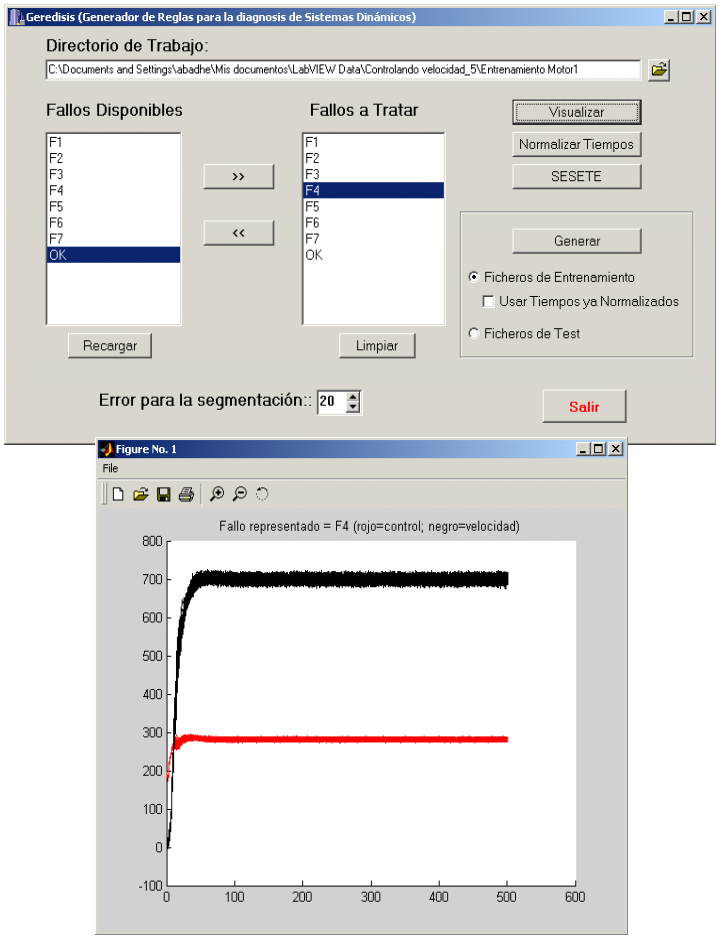

Figura 7.20: Herramienta para la generación de reglas con DISETRA

(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	<-classified as
12						8		(a): class F__OK
	20							(b): class F__F1
		20						(c): class F__F2
			20					(d): class F__F3
				20				(e): class F__F4
3					17			(f): class F__F5
						9	11	(g): class F__F6
						2	18	(h): class F__F7

Tabla 7.12: Casos de test sobre el motor 1 con DISETRA

motor distinto al usado para el entrenamiento. Por supuesto, el nuevo motor tiene características semejantes al usado en el entrenamiento. Los resultados obtenidos con este nuevo motor son los mostrados en la tabla 7.13.

(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	<-classified as
18					2			(a): class F__OK
2	17					1		(b): class F__F1
		20						(e): class F__F2
			20					(f): class F__F3
				20				(g): class F__F4
					20			(c): class F__F5
						20		(d): class F__F6
						2	18	(h): class F__F7

Tabla 7.13: Casos de test sobre el motor 2 con DISETRA

Como puede comprobarse en los resultados la precisión obtenida es bastante buena para la mayoría de los casos. En el caso de los test realizados los peores resultados se obtienen al diagnosticar el comportamiento OK. Esto resulta normal si se tiene en cuenta que esa es la clase que el algoritmo catalogó como *no diferenciable*. Este efecto podría haber sido paliado usando una fase de detección antes de la de diagnosis, lo que hubiese hecho innecesario tener que clasificar dicha clase. Esta teoría cobra más fuerza debido a la clase de que se trata, puesto que los falsos positivos son una medida muy importante en el rendimiento. No se ha querido plantear dicha posibilidad para mostrar al completo los resultados ofrecidos por el algoritmo DISETRA.

Resulta muy curioso comprobar que los resultados son más precisos con el motor con el que no se ha realizado el entrenamiento, incluso para el comportamiento OK.

La confusión más común entre fallos está entre los fallos *F6* y *F7*, pero esto resulta natural, puesto que ambos fallos provocan una evolución del sistema muy similar.

Para comprobar la validez de los resultados se han sometido esos mismos casos de test a la evaluación de las reglas obtenidas por el algoritmo de clasificación C5.0®, obtenidas con el mismo conjunto de entrenamiento y bajo las mismas condiciones se segmentación y normalización. Los resultados arrojados de evaluar los test con dicho conjunto de reglas se muestran en las tablas 7.14 y 7.15.

Puede observarse como la clasificación realizada por DISETRA supera en términos generales a la ofrecida por C5.0, sobre todo en aquellos fallos que DISE-

(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	<-classified as
18	1					1		(a): class F__OK
	20							(b): class F__F1
		12				2	6	(c): class F__F2
			20					(d): class F__F3
				19			1	(e): class F__F4
1					19			(f): class F__F5
						8	12	(g): class F__F6
						1	19	(h): class F__F7

Tabla 7.14: Casos de test sobre el motor 1 con C5.0

(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	<-classified as
15						4	1	(a): class F__OK
10	10							(b): class F__F1
		9				1	10	(c): class F__F2
			20					(d): class F__F3
				19			1	(e): class F__F4
					20			(f): class F__F5
						19	1	(g): class F__F6
							20	(h): class F__F7

Tabla 7.15: Casos de test sobre el motor 2 con C5.0

TRA marca como *totalmente separables*, en los cuales no comente ningún error de clasificación, al contrario de lo que ocurre si evaluamos con las reglas de C5.0. Concretamente, el fallo F2, es siempre diagnosticado perfectamente con DISETRA, mientras que C5.0 falla el 40 % de las veces con el motor 1 y el 55 % de las veces con el motor 2. El fallo F4, también es perfectamente diagnosticado por DISETRA, mientras que C5.0, comete un 5 % de error en ambos motores. Esto no sería significativo, a no ser porque el fallo F4, es muy diferente del fallo F7 con el cual los confunde. Por contra, el comportamiento correcto es diagnosticado con mayor precisión por C5.0, en el motor 1, mientras que DISETRA es más preciso en el motor 2. El motivo de la falta de precisión de DISETRA en este apartado es el ya anteriormente mencionado de la *no diferenciabilidad* de la clase OK.

Para comprobar que las reglas pueden ser evaluadas *on-line*, éstas han sido implementadas en el mismo PC que se usa para recoger la lectura de los sensores, obteniéndose los tiempos de respuesta esperados. La figura 7.21 muestra la evaluación *on-line* del fallo F2.

Figura 7.21: Diagnosis *On-Line* del fallo F2

Capítulo 8

Conclusiones y Trabajo Futuro

8.1. Conclusiones

El presente proyecto de tesis ha desarrollado una completa metodología para la diagnosis de sistemas dinámicos mediante el aprendizaje de modelos proposicionales. Se han presentado los antecedentes del problema, así como el estado actual de la investigación en el campo de la diagnosis automática. A partir de ahí, se ha seleccionado el proceso de $\mathcal{KDD}$ como base para la metodología a aplicar, adaptando sus distintas fases a las necesidades del problema en cuestión. Dentro de dicho proceso, la técnica seleccionada para el aprendizaje han sido los árboles de decisión, puesto que sus propiedades se adaptan convenientemente a los requisitos de coste y velocidad que se necesitaban.

Tras los ensayos realizados con los diferentes modelos, y posteriormente con el sistema real, se demuestra la validez de la metodología propuesta, en el presente proyecto de tesis, para la realización de diagnosis de sistemas dinámicos mediante el aprendizaje de modelos proposicionales.

Se ha conseguido alcanzar un diagnóstico acertado, dentro de unos porcentajes bastante altos, tanto para las simulaciones de los modelos matemáticos, como para los ensayos con el sistema real. Se demuestra de esta forma, la no necesidad del uso de modelos para llevar a cabo la diagnosis, siendo únicamente necesario disponer de los datos experimentales del sistema en aquellos transitorios en los que va a ser monitorizado y para aquellos fallos que se pretenden diagnosticar.

Se ha comprobado, también, como la técnica de segmentación de *trayectorias*, para reducir la dimensionalidad de las mismas, consigue aproximar dichas trayectorias con una cantidad de información más pequeña, eliminando de esta forma toda la información no relevante presente en la misma. La propia técnica de segmentación consigue también hacer un uso eficiente de la derivada de los segmentos, permitiendo, este nuevo atributo, un enriquecimiento del modelo proposicional conseguido posteriormente. Esto también se consigue con la incorporación del resto de atributos calculados sobre la trayectoria. El hecho de añadir estos nuevos atributos provoca un aumento de la dimensión de las trayectorias, que no supone un perjuicio gracias a la reducción conseguida con la segmentación.

Una vez conseguido el modelo proposicional, la simplicidad del mismo permite que su coste de almacenamiento sea muy pequeño. Por su parte, esa misma simplicidad provoca que la evaluación se produzca de manera muy rápida y con un coste computacional muy contenido. Esta cualidad produce dos consecuencias inmediatas:

- Por una parte, el coste contenido de almacenamiento y procesamiento permiten usar elementos computacionales pequeños y baratos, como pueden serlo los microcomputadores, lo que reduce enormemente el coste de implantación del dispositivo de diagnóstico en el sistema.

- Por otra parte, la rapidez con la que puede ser evaluado el modelo proporciona la posibilidad de realizar el diagnóstico en tiempo real, incluso para aquellos sistemas en los que la dinámica del transitorio es muy rápida, como es el caso de los motores empleados en los ensayos. Se ha demostrado con dichos ensayos, la posibilidad real de alcanzar el diagnóstico *on-line*.

Los ensayos realizados, con los motores reales, han demostrado la posibilidad de incorporar el sistema de diagnóstico a dispositivos fabricados en serie, dado que el modelo conseguido con uno o varios de dichos dispositivos puede servir para diagnosticar a todos los de la serie. Esto ha sido comprobado al realizar el diagnóstico de la segunda unidad del motor utilizado en los ensayos, mediante el modelo obtenido con la primera unidad, siendo los resultados alcanzados muy prometedores.

Cuando los fallos a diagnosticar no presentan diferencias, en la lectura ofrecida por los sensores, dichos fallos no son diferenciables. Las técnicas de *reetiquetado* y *clasificación multiple* han demostrado ser una opción a la mejora de dicho problema, porque a pesar de que es imposible alcanzar un diagnóstico acertado, se proporcionan todas aquellos posibles diagnósticos que pueden serlo, sin apostar por ninguno en concreto, lo cual podría producir una perdida de tiempo y un coste adicional al intentar solucionar un fallo que no es el correcto. El diagnóstico múltiple ofrecido puede servir como base a pruebas adicionales que permitan identificar unívocamente el fallo.

La experiencia adquirida al aplicar la metodología a los casos de estudio, reveló una deficiencia a la hora de generar los modelos proposicionales al emplear los clasificadores ortogonales usados. Nos estamos refiriéndoos al valor empleado en los nodos del árbol de clasificación para establecer las separación de las clases. Dicho valor se establece en el valor de separación de una de dichas clases, con lo cual, cualquier nueva observación próxima a dicho valor pero que supere dicho umbral es incorrectamente clasificada. Así pues, para subsanar esto, se ha propuesto un nuevo algoritmo *DISETRA* (sección 6.8) que busca una frontera de separación equidistante a las clases que divide de forma que el problema quede subsanado. Dicho algoritmo no se limita a modificar el valor de división, sino que establece una completa estrategia para conseguirlo, empleando en tal fin las técnicas validadas por la metodología propuesta. Los resultados empíricos demuestran como se mejora en el diagnóstico, sobre todo para fallos cuyas evoluciones están muy próximas entre sí.

8.2. Trabajo Futuro

A partir del trabajo desarrollado, en el presente proyecto de tesis, pueden continuarse una serie de líneas de investigación que permitan ahondar y mejorar las ideas aquí expuestas.

Una tarea inmediata, que puede comenzarse, es la aplicación de las técnicas aquí presentadas a fallos simultáneos de múltiples componentes. Dado que los fallos que se pretenden diagnosticar deben reproducirse a priori, la tarea consistirá en reproducir fallos de multiples componentes y aplicar las técnicas ya presentadas.

Este tipo de fallos son más complejos de diagnosticar que los fallos de un único componente, con lo cual habrá que subsanar los problemas que surjan de dicha complejidad.

También de forma directa, se podrán aplicar las técnicas presentadas a otros sistemas dinámicos, que por su estructura o comportamiento se adecúen a las necesidades requeridas. Estos nuevos sistemas tendrán sus propias características, que harán que surjan nuevos problemas y retos para su investigación.

Se podría extender, también, el diagnóstico al estacionario de los sistemas dinámicos. Puesto que la base de la metodología presentada es la clasificación automática, que require de un conjunto de datos de aprendizaje, se debería, de alguna forma, gestionar el periodo estacionario para poder tener el conjunto de datos de aprendizaje. Por ejemplo, se podría generar una ventana, de un determinado tamaño, que se deslice a la vez que lo hace el estacionario, aplicando la metodología a dicha ventana, con los necesarios ajustes, por supuesto.

Una de las tareas que pueden mejorarse es la búsqueda automática el umbral del error que debería usarse en el algoritmo de segmentación. Hasta ahora, la experiencia y la intuición marcan la colocación de dicho umbral, pero sería necesario encontrar un método automático de colocarlo. En ese mismo sentido, la forma en la que se realiza el cálculo del error puede ser también un punto de partida para la mejora del proceso de segmentación.

Otro de los problemas que quedan pendientes de subsanar es el problema de los fallos no previstos. Para que la metodología pueda encontrar el fallo, éste debe haberse previsto con anterioridad. Ante la presencia de un fallo no previsto, se podría buscar una alternativa que permitiese la realimentación de dicho fallo para que fuese detectada en un ocurrencia futura.

La tarea de clasificación, en sí misma, es otro de los posibles puntos de partidas para futuras investigaciones. Se podrían, posiblemente, mejorar los resultados usando técnicas como el *stacking* o el *cascading* (sección 5.6), siempre y cuando se salve el problema de la velocidad de evaluación posterior del conjunto de reglas generadas, que es uno de los objetivos pretendidos.

El algoritmo *DISETRA* (sección 6.8) es una idea que puede seguir mejorándose en múltiples frentes. Por una parte, cuando dos conjuntos de trayectorias no son separables, ni totalmente ni individualmente con otras, el algoritmo, simplemente,

las marca como 'no separables'. Esta situación puede ser mejorada si se busca la separación óptima, que aunque no consiga una regla que divida totalmente las dos clases, sea capaz de distinguir una mayoría de casos, dejando el umbral de solapamiento como 'no separable' o que establezca un grado de pertenencia según los casos que no se puedan separar.

Por otra parte, en la presente memoria se ha dejado abierta la posibilidad de generar distintas reglas de separación del algoritmo *DISETRA*, dependiendo del criterio que se pretenda seguir. El estudio de dichos criterios, la búsqueda de la regla optima, o la combinación de más de una posibilidad, es algo que está pendiente de definir. Incluso cabe la posibilidad de no generar una regla única, sino una combinación ponderada de las distintas opciones ofertadas por el algoritmo.

Toda la información que se ha manejado ha sido información cuantitativa. El uso de información cualitativa junto con la ya existente abre nuevas líneas de investigación donde se podrán aplicar técnicas provenientes de otros campos.

Finalmente, el diagnóstico realizado ha sido centralizado. Cuando la ubicación de la información o la estructura del sistema a diagnosticar no permitan realizar esta diagnosis, se deberá adaptar la solución para poder realizar diagnosis distribuida.

En definitiva, estas son algunas de las tareas que de forma inmediata pueden abordarse a partir del trabajo ya realizado, pero, por supuesto que no son las únicas, y cada una de ellas podrá llevar a nuevos problemas que necesiten nuevas soluciones.

Apéndice A

Hardware y Software Desarrollado

A.1. Hardware desarrollado

Para poder llevar a cabo los experimentos mostrados con el sistema expuesto en el apartado 7.4.1, ha sido necesario desarrollar parte del hardware utilizado. El diagrama de bloques de la figura A.1 muestra los elementos que permiten controlar el motor de corriente continua de excitación independiente *ALECOP AL 506; 220V/0.61 KW* mostrado en la figura A.2.

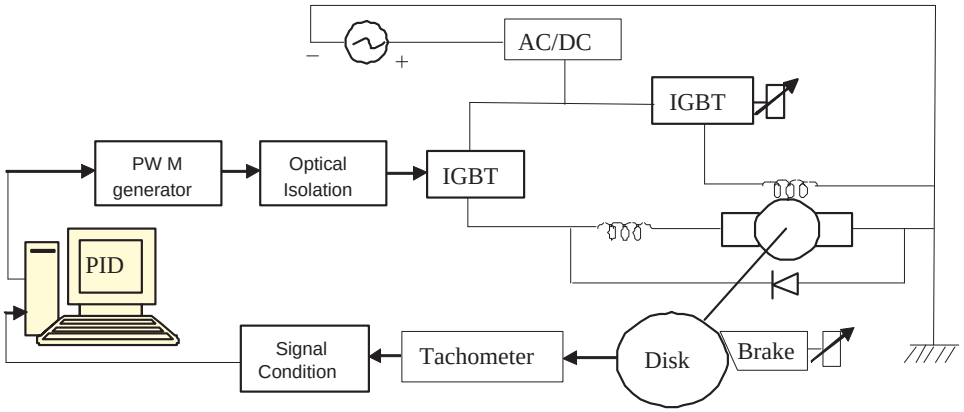

Figura A.1: Diagrama de bloques

El hardware desarrollado específicamente se encarga de las siguientes funciones:

- Generación de la señal PWM.
- Aislamiento óptico.

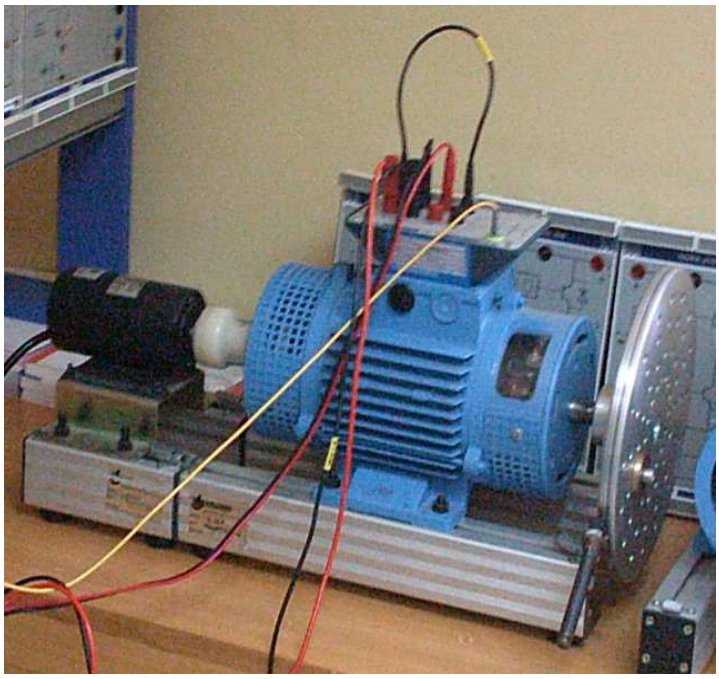

Figura A.2: Motor ALECOP AL 506; 220V/0.61 KW

- Acondicionamiento de la tensión de alimentación del rotor en base a la velocidad deseada.
- Acondicionamiento de la señal del tacómetro.

En la figura A.3 se muestra el diagrama del circuito desarrollado con tales fines, y en la figura A.4 se puede observar el circuito propiamente dicho.

El circuito desarrollado dispone de un generador *PWM SG3524N* que se encarga de generar una onda cuyo ciclo de servicio está modulado en base a la señal analógica que recibe de la tarjeta de adquisición de datos a través del canal 1. La frecuencia de dicha señal se fija mediante una célula *RC*. La señal PWM generada debe aplicarse a la base del IGBT que controla el rotor del motor. Para conseguir un aislamiento eléctrico imprescindible para controlar cargas de potencia, se utiliza un optoacoplador *4N25*. Además, para eliminar posibles conmutaciones indeseadas debidas al ruido, se emplea una puerta con histéresis *74H132*. La salida de dicha puerta ofrece una señal PWM limpia para excitar el IGBT. La necesidad de refrigeración de los elementos electrónicos de potencia (IGBT y diodo de libre circulación) requiere de dos ventiladores para cumplir dicha función.

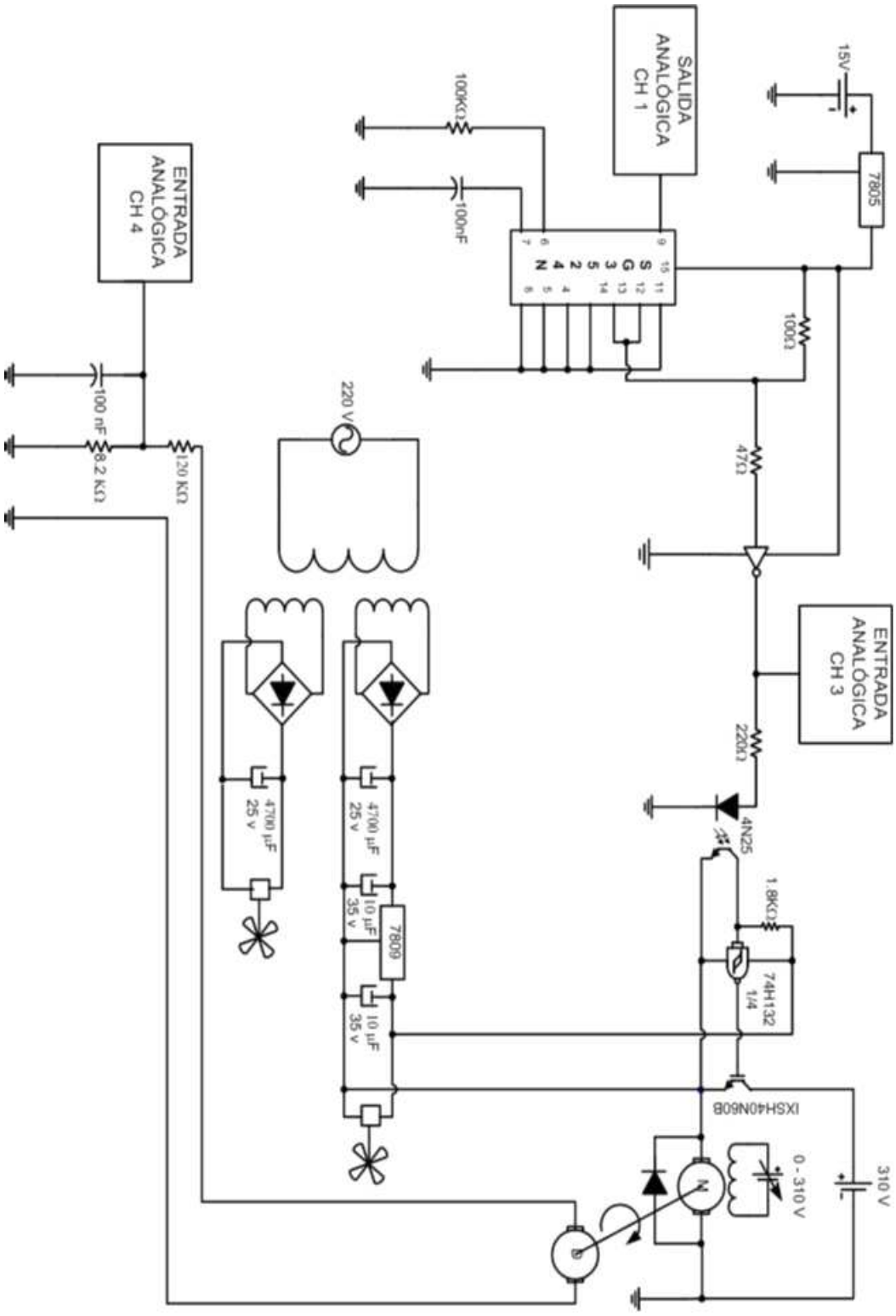

Figura A.3: Diagrama del circuito desarrollado

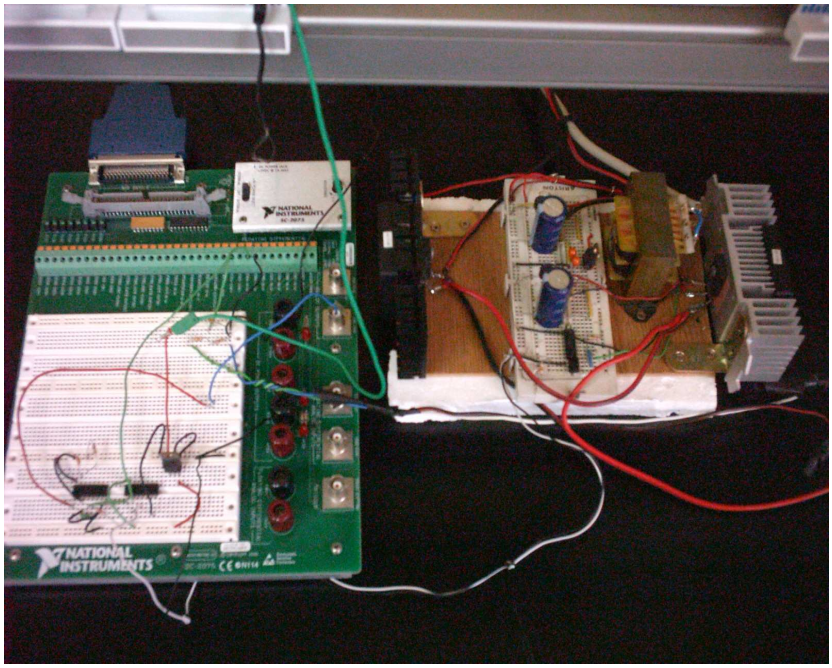

Figura A.4: Circuito desarrollado

La velocidad de giro del motor es la señal que deseamos recoger para nuestros propósitos. El problema que plantea la recogida de dicha señal es el nivel de tensión ofrecido por el tacómetro, que es muy superior al aceptado por la entrada analógica de la tarjeta. Con el fin de acondicionar dicha señal, se usa un divisor de tensión y un filtro. El divisor está compuesto por dos resistencias con los valores adecuados y el filtrado se realiza mediante un condensador. El resultado es una señal proporcional a la velocidad y de un nivel adecuado para poder ser capturada por la tarjeta.

A.2. Software desarrollado

Por supuesto, para la realización de los distintos experimentos, ha sido necesario desarrollar el software adecuado que implemente las funciones y algoritmos que se han requerido a lo largo del desarrollo de la presente tesis. Los distintos algoritmos y rutinas necesarias han sido elaborados en diversos lenguajes de programación, dependiendo de las herramientas utilizadas en cada momento. Los más utilizados han sido:

- ANSI C
- Borland C++ Builder
- MATLAB
- LABVIEW

La mayoría de las veces, este software ha estado compuesto por programas específicos que manipulaban la información con objeto de conseguir los fines deseados, sin que ello conllevara ningún tipo de entorno e interacción con el usuario. En la fase final de la tesis, dichos algoritmos han sido reunidos en una herramienta a la que se ha dado el nombre de *Geredisis* que posibilita la manipulación de los datos y la obtención de los resultados en un entorno gráfico que permite la interacción con el usuario y la selección de aquellas opciones deseadas. Esta herramienta ha sido programada con *Borland C++ Builder*, reutilizando, en la medida de lo posible, parte del código implementado mediante MATLAB. La figura A.5muestra alguna de las posibilidades de dicha herramienta.

La herramienta *Geredisis* permite, de forma amigable, seleccionar el directorio de trabajo donde se encuentran los distintos ficheros con los datos de los distintos comportamientos a diagnosticar. Una vea seleccionado el directorio, se muestran en una ventana todos los comportamientos disponibles en dicho directorio. Dichos comportamientos pueden ser seleccionados para su tratamiento, ofreciéndose la posibilidad de visualizar gráficamente la evolución temporal de las variables de cada uno de los comportamientos seleccionados. Con los comportamientos seleccionados se pueden generar los ficheros de entrenamiento necesarios para generar las reglas con la herramienta de aprendizaje C5.0 en el formato adecuado. Es posible también, generar los ficheros de test, para validar los resultados.

En la herramienta *Geredisis* se incorpora también la posibilidad de generar las reglas obtenidas por SESETE, según se expuso en el apartado 6.8.

Por su parte, la implantación del sistema de diagnóstico, una vez generadas las reglas, requiere de un software que permita la interacción con el hardware. Para tal fin, se ha utilizado la herramienta de desarrollo Labview, desarrollando por una parte, la aplicación que recoge los datos de entrenamiento en las diferentes situaciones de diagnóstico posible, y por otra, una aplicación capaz de implementar las reglas de clasificación obtenidas y diagnosticar el funcionamiento del motor

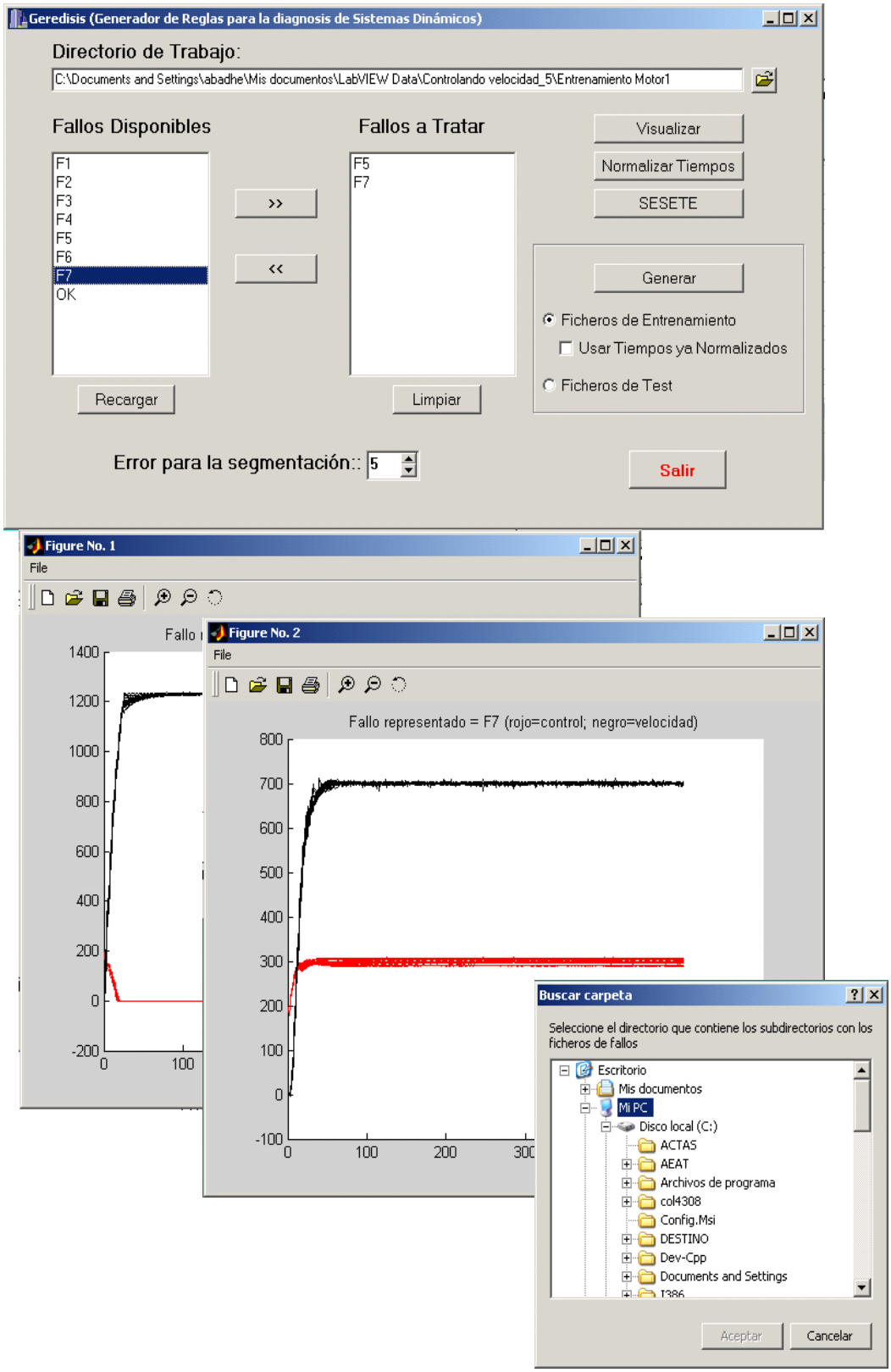

Figura A.5: Alguna de las posibilidades de la herramienta *Geredisis*

en base a las lecturas tomadas. Las imágenes A.7y A.6 muestran el diagrama de bloques y el panel de control de la aplicación de captura de datos respectivamente. Por su parte, en las figura A.8 se muestra el panel de control de la aplicación de diagnóstico.

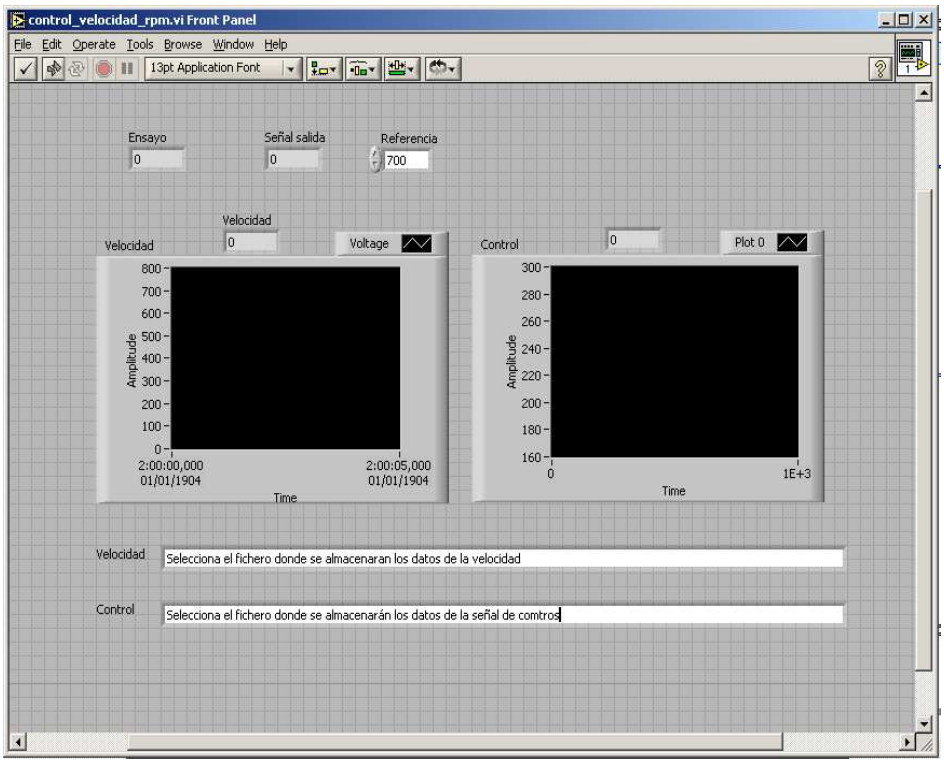

Figura A.6: Panel de Control de la aplicación Labview para la recogida de datos

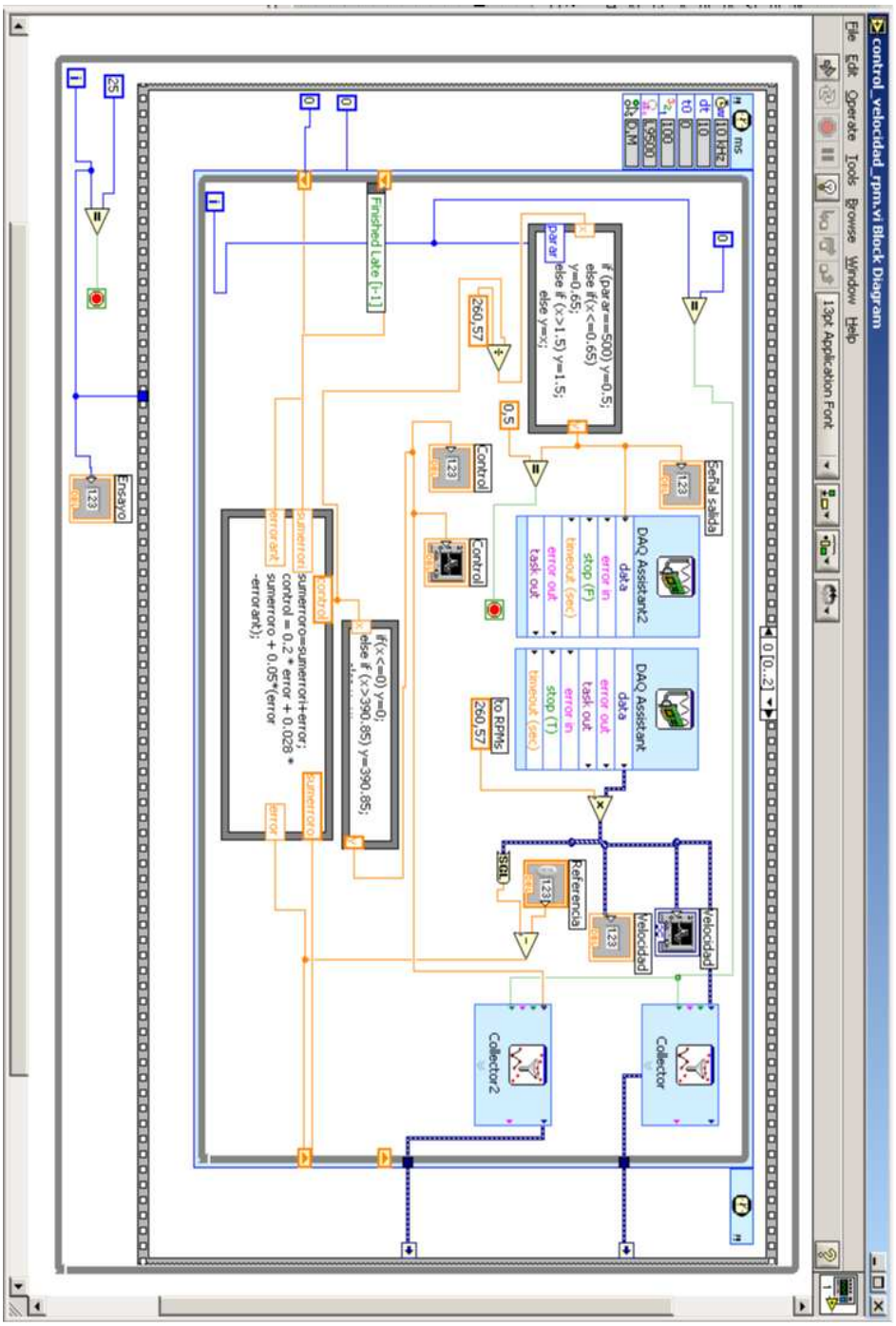

Figura A.7: Diagrama de bloques de la aplicación Labview para la recogida de datos

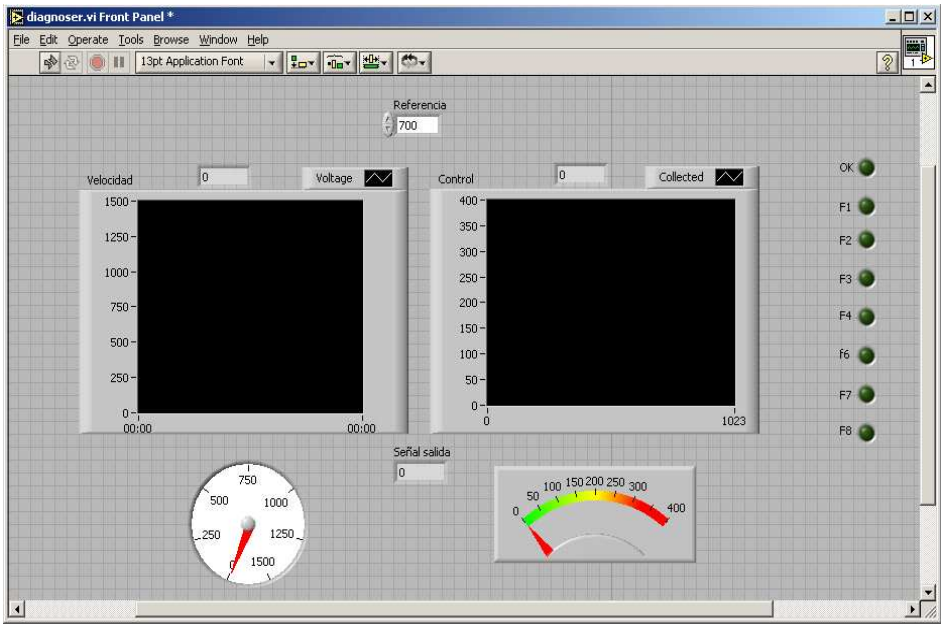

Figura A.8: Panel de Control de la aplicación Labview para el diagnóstico

Bibliografía

[Aamodt and Plaza, 1994] A. Aamodt and E. Plaza. Case-based reasoning: Fundational issues, methodological variations, and system approaches. *AI Communications*, 7(1):39–59, 1994.

[Abad *et al.*, 2002] Pedro J. Abad, Antonio J. Suárez, Rafael M. Gasca, and Juan A. Ortega. Using supervised learning techniques for diagnosis of dynamic systems. In *Proceedings of the 13th International Workshop on Principles of Diagnosis*, Semmering, Austria, 2002. M.Stumptner & F.Wotawa Ed.

[Abad *et al.*, 2005] Pedro J. Abad, Antonio J. Suarez, Rafael M. Gasca, and Juan A. Ortega. Diagnosis of a chopper-controlled dc motor by boosting. In *CIMCA '05: Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce Vol-2 (CIMCA-IAWTIC'06)*, pages 568–575, Washington, DC, USA, 2005. IEEE Computer Society.

[Agrawal *et al.*, 1998] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 94–105, 1998.

[Aguado and Aguilar, 1999] J. C. Aguado and J. Aguilar. A mixed qualitative-quantitative classification method applied to diagnosis. In *Proceedings of the 10th International Workshop on Principles of Diagnosis*, 1999.

[Aha *et al.*, 1991] D. Aha, D. Kibler, and M. Albert. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.

[Arabie *et al.*, 1996] P. Arabie, L.J. Hubert, and G. De Soete. Clustering and classification. *World Scientific*, 1996.

[Balakrishnan and Jonavar, 1998] K. Balakrishnan and V. Jonavar. Intelligent diagnosis systems. *Journal of Intelligent Systems*, 8, 1998.

[Bennett, 1985] J. S. Bennett. A knowledge-based system for acquiring the conceptual structure of an expert system. *Journal of Automated Reasoning*, 1(1):49–74, 1985.

[Biswas *et al.*, 1999] G. Biswas, R. Kapadia, and X. W. Yu. Combined qualitative-quantitative steady state diagnosis ofcontinuous-valued systems. *IEEE Transactions on systems, man, and cybernetics*, 1, 1999.

[Bocaniala and da Costa, 2005] C. D. Bocaniala and J. Sa da Costa. Novel methodology for partitioning complex systems for fault diagnosis purposes. In *Preprints of the 16th IFAC World Congress*, Praha, Czech Republic, 2005.

[Bousson and Travé-Massuyes, 1993] K. Bousson and L. Travé-Massuyes. Fuzzy causal simulation in process engineering. In *Proceedings of the 13th International Joint Conference On Artificial Intelligence*, Chambery, 1993.

[Bousson and Trave-Massuyes, 1992] K. Bousson and L. Trave-Massuyes. A computational causal model for process supervision. Technical Report 92147, LAAS-CNRS, Toulouse, France, 1992.

[Bousson, 1993] K. Bousson. *Raisonnement Causal pour la Supervision de Processus Basee sur des modeles.* PhD thesis, LAAS-CNRS, Toulouse, 1993.

[Bregon *et al.*, 2006] Anibal Bregon, Belarmino Pulido, M. Aranzazu Simon, Isaac Moro, Oscar Prieto, Juan J. Rodriguez, and Carlos Alonso. Focussing fault localization in model-based diagnosis with case-based reasoning. In *Proceeding of the DX06*, Burgos (Spain), 2006.

[Breiman *et al.*, 1984] L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone. *Classification and Regression Trees.* Wadsworth International Group, Belmont, CA, 1984.

[Breuker and de Velde, 1994] J. Breuker and W. Van de Velde. *CommonKADS library for expertise modelling.* 1994.

[Brown *et al.*, 1982] J. S. Brown, R. Burton, and J. de Kleer. *Pedagogical, Natural Language and knowledge engineering issues in SOPHIE I, II y III.* Intelligent Tutoring System. Academic Press, NY, 1982.

[Brusoni *et al.*, 1998] V. Brusoni, L. Console, P. Terenziani, and D. T. Dupré. A spectrum of definitions for temporal model-based diagnosis. *Artificial Intelligence*, 102:39–79, 1998.

[Buchanan and Shortliffe, 1984a] Bruce G. Buchanan and Edward H. Shortliffe. Rule based expert system. the mycin experiments of the stanford heuristics programming project. *Artificial Intelligence*, 1984.

[Buchanan and Shortliffe, 1984b] Bruce G. Buchanan and Edward H. Shortliffe. Rule based expert system. the mycin experiments of the stanford heuristics programming project. *Artifial Intelligence. Addison Wesley, Reading*, 1984.

[Calado *et al.*, 2001] J.M.F. Calado, J. Korbicz, K. Patan, R.J. Patton, and J.M.G. Sa da Costa. Soft computing approaches to fault diagnosis for dynamic systems. *European Journal of Control*, 7(3):169–208, 2001.

[Cascio *et al.*, 1999] F. Cascio, L. Console, L. Guagliumi, M. Osella, A. Panati, S. Sottano, and D. Theisider Dupré. Generating on-board diagnostics of dynamic automotive systems based on qualitative models. In *Proceedings of the QR99*, Loch Awe, Scotland, 1999.

[Casimir *et al.*, 2006] R. Casimir, E. Boutleux, G. Clerc, and A. Yahoui. The use of feature selection and nearest neighbors rule for fault diagnosis in induction motors. *Engineering Applications of Artificial Intelligence*, 19(2):169–177, 3 2006.

[Catlett, 1991] J. Catlett. On changing continuous attributes into ordered discrete attributes. In *Proceedings of European Working Session on Learning*, pages 164–178, Berlin, Alemania, 1991. Springer Verlag.

[Cestnik *et al.*, 1987] G. Cestnik, I. Kononenko, and I. Bratko. *Assistant 86: A knowledge acquisition tool for sophisticated users.* Bratko, I., & Lavrac, N. (Eds.), Progress in Machine Learning. Sigma Press, 1987.

[Chaisaowong *et al.*, 2007] Kraisorn Chaisaowong, Patrick Jäger, Stefan Vogel, Achim Knepper, Thomas Kraus, and Til Aach. Computer-assisted diagnosis for early stage pleural mesothelioma: Towards automated detection and quantitative assessment of pleural thickenings from thoracic ct images. *Methods of Information in Medicine*, 46(3):324–331, 2007.

[Chan and Fu, 1999] Kin-Pong Chan and Ada Wai-Chee Fu. Efficient time series matching by wavelets. In *ICDE*, pages 126–133, 1999.

[Chan *et al.*, 1999] C. W. Chan, K. C. Cheung, H. Y. Zhang, and Y. Wang. Fault detection of dc-motors using non-linear observer based on current b-spline neurofuzzy network. In *Proceedings of the 14th IFAC World Congress*, volume B, Beijing, China, 1999.

[Cheeseman and Stutz, 1996] P. Cheeseman and J. Stutz. Bayesian classification (autoclass): Theory and results. In *Advances in Knowledge Discovery and Data Mining*, pages 153–180. AAAI/MIT Press, 1996.

[Chen and Wei, 2002] G. Chen and Q. Wei. Fuzzy association rules and the extended mining algorithms. *Information Sciences*, 147:201–228, 2002.

[Chen and Wotawa, 2006] Rong Chen and Franz Wotawa. Diagnosing program errors with light-weighted specifications. In *IEA/AIE*, pages 639–649, 2006.

[Chen *et al.*, 1996] M.S. Chen, J. Han, and P.S. Yu. Data mining: An overview from database perspective. *IEEE Transactions on knowledge and Data Engineering*, 8(6):866–883, 1996.

[Chen *et al.*, 2004] M. Chen, A.X. Zheng, J. Lloyd, M.I. Jordan, and E. Brewer. Failure diagnosis using decision trees. In *Proceedings of the International Conference on Autonomic Computing*, pages 36– 43, California Univ., Berkeley, CA, USA, May 2004.

[Chittaro *et al.*, 1992] L. Chittaro, G. Guida, C. Tasso, and E. Toppano. Developing diagnostic applications using multiple models: the role of interpretative knowledge. In A.Stefanini (Eds.) G.Guida, editor, *Industrial Applications of Knowledge-based Diagnosis*, pages 219–263. Elsevier, Amsterdam, NL, 1992.

[Clancey, 1984] W. J. Clancey. *Acquiring, representing, and evaluating a competence model of diagnosis.* The Nature of Expertise, in preparation, 1984.

[Console and Dressler, 1999] L. Console and O. Dressler. Model-based diagnosis in the real world lessons learned and challenges remaining. In *Proceedings of the IJCAI'99*, Stocolmo, 1999.

[Console and Torasso, 1991] L. Console and P. Torasso. A spectrum of logical definitions of model based diagnosis. *Computational Intelligent*, 1991.

[Console *et al.*, 1989] L. Console, D. Theisider Dupre, and P. Torasso. A theory of diagnosis for incomplete causal models. In *Proceedings of the 10th international Joint conference on Artificial Intelligent*, USA, 1989.

[Console *et al.*, 1994] L. Console, L. Portinale, D. Theseider Dupré, and P. Torasso. Diagnosing time-varying misbehavior: an approach based on model descomposition. *Annals of Mathematics and Artificial Intelligence, special issue on Model-based diagnosis*, 1994.

[Console *et al.*, 2000] L. Console, C. Picardi, and M. Ribaudo. Diagnosis and diagnosability using pepa. In *Proceedings of the 14th European Conference in Artificial Intelligence.* IOs Press, 2000.

[Cover and Thomas, 1991] T.M. Cover and J.A. Thomas. *Elements of Information Theory.* Series in Telecommunications. John Wiley and Sons edition, 1991.

[Dague P. and Raiman, 1992] Deves P. Dague P. and O. Raiman. Troubleshooting: when modelling is the trouble. In Console L. Hamscher, W. and J. (eds) de Kleer, editors, *Readings in Model Based Diagnosis.* Morgan Kaufmann, 1992.

[Dague, 1994] P. Dague. Model based diagnosis of analog electronic circuits. *Annals of Mathematics and Artificial Intelligent, special issues in model based diagnosis*, 1994.

[Darwiche, 1995] A. Darwiche. Model-based diagnosis using causal networks. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 1995.

[Dash *et al.*, 2003] Sourabh Dash, Raghunathan Rengaswamy, and Venkat Venkatasubramanian. Fuzzy-logic based trend classification for fault diagnosis of chemical processes. *Computers and Chemical Engineering*, 27(3):347–362, March 2003.

[Davids *et al.*, 2006] Daniel Davids, Siddhartha Datta, Arindam Mukherjee, Bharat Joshi, and Arun Ravindran. Multiple fault diagnosis in digital microfluidic biochips. *J. Emerg. Technol. Comput. Syst.*, 2(4):262–276, 2006.

[Davis, 1984] R. Davis. Diagnostic reasoning based on structure and behavior. *Artif. Intell.*, 24(1-3):347–410, 1984.

[Davis, 1990] R. Davis. Diagnosis via causal reasoning: Paths of interaction and the locality principle. In D. S. Weld and J. de Kleer, editors, *Readings in Qualitative Reasoning about Physical Systems*, pages 535–541. Kaufmann, San Mateo, CA, 1990.

[de Kleer *et al.*, 1992] J. de Kleer, A. Mackworth, and R. Reiter. Characterizing diagnoses and systems. *Artificial Intelligence*, 56:197–222, 1992.

[de Miguel and Blazquez, 2005] Luis J. de Miguel and L. Felipe Blazquez. Fuzzy logic-based decision-making for fault diagnosis in a dc motor. *Engineering Applications of Artificial Intelligence*, 18(4):423–450, 6 2005.

[Dehaspe and Toivonen, 1999] Luc Dehaspe and Hannu Toivonen. Discovery of frequent datalog patterns. *Data Mining and Knowledge Discovery*, 3(1):7–36, 1999.

[DeKleer and Williams, 1987] J. DeKleer and B. Williams. Diagnosing multiple faults. *Artificial Intelligence*, 32:97–130, 1987.

[Dempster *et al.*, 1977] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, B, 39:1–38, 1977.

[Dietterich, 2000a] Thomas G. Dietterich. Ensemble methods in machine learning. *Lecture Notes in Computer Science*, 1857:1–15, 2000.

[Dietterich, 2000b] Thomas G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40(2):139–157, 2000.

[Dougherty *et al.*, 1995] J. Dougherty, R. Kohavi, and M. Sahami. Supervised and unsupervised discretization of continuous features. In *Proceedings of the 12th International Conference on Machine Learning*, pages 194–202, Los Altos, CA, Morgan Kaufmann, 1995.

[Dressler and Freitag, 1994] O. Dressler and H. Freitag. Prediction sharing across time and contexts. In *Proceedings of the 8th. Interntation Workshop on Qualitative Reasoning about Plysical Systems(QR'94)*, Japón, 1994.

[Dressler and Struss, 1994] O. Dressler and P. Struss. Model-based diagnosis with the default-based diagnosis engine: effective control strategies that work in practice. In *Proceedings of the 11th European Conference on Artificial Intelligence (ECAI-94)*, pages 677–681, 1994.

[Dressler and Struss, 1996] O. Dressler and P. Struss. The consistency-based approach to automated diagnosis of devices. In Editor Gerhard Brewka, editor, *Principles of Knowledge Representation*. CSLI Publications, Standfor, 1996.

[Dressler, 1996] O. Dressler. On-line diagnosis and monitoring of dynamic systems based on qualitative models and dependency-recording diagnosis engines. In *Proceedings of the 12th. European Conference on Artificial Intelligence(ECAI-96)*, pages 461–465, 1996.

[Duda and Hart, 1973] Richard O. Duda and Peter E. Hart. *Pattern classification and scene analysis*. Wiley, New York, 1973.

[Dvorak and Kuipers, 1989] D. Dvorak and B. Kuipers. Model based monitoring of dynamic systems. In *Proc. of the 11th IJCAI*, pages 1238–1243, Detroit, MI, 1989.

[Efron, 1983] B. Efron. Estimating the error rate of a prediction rule: improvements on cross-validation. *Journal of American Statistical Association*, 78(382):316–331, 1983.

[Fayyad and Irani, 1993] U.M. Fayyad and K.B. Irani. Multi-interval discretization of continuous valued attributes for classification learning. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pages 1022–1027, Morgan Kaufmann, 1993.

[Fayyad *et al.*, 1996] U.M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery in databases. *Ai Magazine*, 17:37–54, 1996.

[Feng, 1992] C. Feng. Inducing temporal fault dignostic rules from a qualitative model. In S. Muggleton, editor, *Inductive Logic Programming*. Academic Press, London, 1992.

[Filippetti *et al.*, 1992] F. Filippetti, M. Martelli, G. Francesschini, and C. Tassoni. Development of expert system knowledge base to on-line diagnosis of rotor electrical faults of induction motors. In *Proceedings of the 7th IEEE-IAS Annu. Meeting*, Oct 1992.

[Forbus, 1990a] K. D. Forbus. The qualitative process engine. In D. S. Weld and J. de Kleer, editors, *Readings in Qualitative Reasoning about Physical Systems*, pages 220–235. Kaufmann, San Mateo, CA, 1990.

[Forbus, 1990b] K. D. Forbus. Qualitative process theory. In D. S. Weld and J. de Kleer, editors, *Readings in Qualitative Reasoning about Physical Systems*, pages 178–219. Kaufmann, San Mateo, CA, 1990.

[Forbus, 1992] K. D. Forbus. Pushing the edge of the (qp) envelope. In B. Faltings and P. Struss, editors, *Recent Advances in Qualitative Physics*, pages 245–261. MIT Press, Cambridge, MA, 1992.

[Forrester, 1968] J. W. Forrester. *Principles of systems*. Pegasus Communications, 1968.

[Frank, 1987] P. Frank. Fault diagnosis in dynamic systems via state estimation: a survey. In Singh Tzafestas and Schmidt(Eds.), editors, *System Fault Diagnostics Reliability and Related Knowledge-based Approaches*. 1987.

[Frank., 1996] M. P. Frank. Analytical and qualitative model-based fault diagnosis -a survey and some new results. *European Journal of Control*, 2, 1996.

[Frawley *et al.*, 1992] W.J. Frawley, G. Piatetsky-Shapiro, and C. J. Matheus. Knowledge discovery in databases - an overview. *Ai Magazine*, 13:57–70, 1992.

[Freund and Schapire, 1996] Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *International Conference on Machine Learning*, pages 148–156, 1996.

[Friedman *et al.*, 1997] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian networks classifiers. *Machine Learning*, (29):131–136, 1997.

[Fuente, 2001] M. J. Fuente. Detección y diagnóstico de fallos usando redes neuronales. In *Actas de las I Jornadas de Diagnosis, Razonamiento Cualitativo y Sistemas Socioeconómicos*, Valladolid, 2001. Carlos Alonso y Juan Antonio Ortega, Editores.

[Garcez *et al.*, 1997] A. Garcez, G. Zaverucha, and V. Silva. Applying the connectionist inductive learning and logic programming system to power system diagnosis, 1997.

[Garcia *et al.*, 2000] F. Javier Garcia, Virginia Izquierdo, Luis J. de Miguel, and Jose R. Peran. Fault-diagnostic system using analytical fuzzy redundancy. *Engineering Applications of Artificial Intelligence*, 13(4):441–450, 8 2000.

[García *et al.*, 2005] E. García, A. Correcher, F. Morant, E. Quiles, and R. Blasco. Modular fault diagnosis based on discrete event systems. *Discrete Event Dynamic Systems*, 15(3):237–256, 2005.

[Gasca *et al.*, 2001] R. M. Gasca, J. A. Ortega, M. Toro, and F. De la Rosa. Diagnosis dirigida por restricciones simbólicas para modelos polinómicos. In *Actas de las I Jornadas de Trabajo sobre Diagnosis, Razonamiento Cualitativo y Sistemas Socioeconómicos*, Valladolid, 2001. Carlos Alonso y Juan Antonio Ortega, Editores.

[Gates, 1972] G.E. Gates. The reduced nearest neighbour rule. *IEEE Transactions on Information Theory*, 18:431–433, 1972.

[Ge and Smyth, 2001] X. Ge and P. Smyth. Segmental semi-markov models for endpoint detection in plasma etching, 2001.

[Ge *et al.*, 2004] Ming Ge, R. Du, Guicai Zhang, and Yangsheng Xu. Fault diagnosis using support vector machine with an application in sheet metal stamping operations. *Mechanical Systems and Signal Processing*, 18(1):143–159, Jaunary 2004.

[Genesereth, 1982] M. Genesereth. Diagnosis using hierarchical design models. In *Proceedings of the 2nd National Conference on Artificial Intelligence*, Pittsburgh PA, 1982.

[Genesereth, 1984] M. Genesereth. The use of design descriptions in automated diagnosis. *Artificial Intelligence*, 24:411–436, 1984.

[Gertler, 1991] J. Gertler. Analytical redundancy methods in failure detection and isolation in complex plants. In *Proceedings of the IFAC/IMACS Symposium SAFEPROCESS-91*, 1991.

[Group, 1999] SIGLA Group. Modals intervals. basic tutorial. In *Proceedings of the Workshop MISC99*, Girona, 1999.

[Guida and Tasso, 1995] G. Guida and C. Tasso. *Design and development of knowledge-based systems: from life cycle to methodology.* John Wiley & Sons, Inc., New York, NY, USA, 1995.

[Hajiaghajani *et al.*, 2004] Massoud Hajiaghajani, Hamid A. Toliyat, and Issa M. S. Panahi. Advanced fault diagnosis of a dc motor. *IEEE Transctions On Energy Conversion*, 19(1), March 2004.

[Hampson and Volper, 1986] S. Hampson and D. Volper. Linear function neurons: Structure and training. *Biological Cybernetics*, 53:203–217, 1986.

[Hamscher, 1991] W. C. Hamscher. Modeling digital circuits for troubleshooting. *Artificial Intelligence*, 51:223–271, 1991.

[Hart, 1967] P. Hart. The condensed nearest neighbor rule. *IEEE Transactions on Information Theory*, 14(3):515–516, 1967.

[Harutunyan *et al.*, 2006] Gurgen Harutunyan, Valery A. Vardanian, and Yervant Zorian. Minimal march-based fault location algorithm with partial diagnosis for all static faults in random access memories. In *DDECS- Workshop on Design and Diagnostics of Electronic Circuits and Systems*, pages 262–267, 2006.

[He and Shi, 2002] F. He and W. Shi. Wpt-svms based approach for fault detection of valves in reciprocating pumps. In *Proceedings of the American Control Conference*, 2002.

[Heckerman *et al.*, 1992] D. Heckerman, E. Horvitz, and B.Ñathwani. Toward normative expert systems: Part i. the pathfinder project. *Methods of Information in Medicine*, 31:90–105, 1992.

[Heiming and Lunze, 1997] B. Heiming and J. Lunze. Parallel knowledge-based process diagnosis applied to a local power station plant. In *Proceedings of the Safeprocess*, 1997.

[Heller and Struss, 2001] U. Heller and P. Struss. G+de. the generalized diagnosis engine. In *Proceedings of the 12th International Workshop on Principles of Diagnosis*, Sansicario, 2001.

[Hernández Orallo *et al.*, 2004] José Hernández Orallo, María J. Ramirez Quintana, and Cesar Ferri Ramírez. *Introducción a la Minería de Datos.* Pearson Educación, 2004.

[Hinneburg and Keim, 1998] A. Hinneburg and D. Keim. An efficient approach to clustering large multimedia databases with noise. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98)*, New York City, USA, 1998. AAAI Press.

[Holte, 1993] R.C. Holte. Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 11:63–90, 1993.

[Horvitz and Heckerman, 1986] E. J. Horvitz and D. E. Heckerman. The inconsistent use of measures of certainty in artificial intelligence research. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, Philadelphia, 1986.

[Hunt *et al.*, 1966] E.B. Hunt, J. Marin, and P.J. Stone. *Experiments in Induction.* Acedemic Press, 1966.

[Hutter and Dearden, 2003] F. Hutter and R. Dearden. Efficient on-line fault diagnosis for nonlinear systems, 2003.

[Inmon, 1992] W.H. Inmon. Eis and the data warehouse: a simple approach to building an effective foundation for eis. *Database Programming & Design*, 5(11):70–73, 1992.

[Isermann, 1993] R. Isermann. Fault diagnosis of machines via parameter estimation and knowledge processing: tutorial paper. *Automatica*, 29(4):815–835, 1993.

[Isermann, 1997] R. Isermann. Supervision, fault detection and fault-diagnosis -an introduction. *Control Engineering Practice*, 5(5), 1997.

[Isermann, 2006] R. Isermann. *Fault Diagnosis Systems. An introduction from fault diagnosis to fault tolerance.* Springer, 2006.

[Jain and Dubes, 1988] A.K. Jain and C. Dubes. *Algorithms for Clustering Data.* Prentice Hall, 1988.

[Janikow, 1998] C. Z. Janikow. Fuzzy decision trees: issues and methods. *IEEE transactions on Systems, Man, and Cybernetics*, 28(1):1–14, 1998.

[Karpenko and Sepehri, 2002] M. Karpenko and N. Sepehri. Neural network classifiers applied to condition monitoring of a pneumatic process valve actuator. *Applications of Artificial Intelligence*, 15(3-4):273–283, 10 2002.

[Kaufman L., 1990] Rousseeuw P.J. Kaufman L. *Finding Groups in Data: An Introduction to Cluster Analysis.* John Wiley & Sons, 1990.

[Keogh *et al.*, 2001a] E. Keogh, S. Chu, D. Hart, and M. Pazzani. An online algorithm for segmenting time series. *IEEE Intl. Conf. Data Mining*, pages 289–296, May 2001.

[Keogh *et al.*, 2001b] Eamonn J. Keogh, Kaushik Chakrabarti, Michael J. Pazzani, and Sharad Mehrotra. Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and Information Systems*, 3(3):263–286, 2001.

[Khoo *et al.*, 2000] L. P. Khoo, C. L. Ang, and J. Zhang. A fuzzy-based genetic approach to the diagnosis of manufacturing systems. *Engineering Applications of Artificial Intelligence*, 13(3):303–310, 6 2000.

[Kira and Rendell, 1992] K. Kira and L.A. Rendell. A practical approach to feature selection. In *Proceedings of the Ninth International Workshop on Machine Learning*, pages 249–256, Aberdeen, Scotland: Morgan Kaufmann, 1992.

[Kleer and B. C. Williams,] J. De Kleer and YEAR = B. C. Williams, TITLE = Diagnosis with Behavioral Modes.

[Kleer and Brown, 1990a] J. De Kleer and J. S. Brown. A qualitative physics based on confluences. In D. S. Weld and J. de Kleer, editors, *Readings in Qualitative Reasoning about Physical Systems*, pages 88–126. Kaufmann, San Mateo, CA, 1990.

[Kleer and Brown, 1990b] J. De Kleer and J. S. Brown. Theories of causal ordering. In D. S. Weld and J. de Kleer, editors, *Readings in Qualitative Reasoning about Physical Systems*, pages 646–660. Kaufmann, San Mateo, CA, 1990.

[Kleer, 1976] J. De Kleer. Local methods of localizing faults in electronic circuits. Technical Report AIM-394, MIT Artificial Intelligent Lab, Cambridge, MA, 1976.

[Kleer, 1990] J. De Kleer. The origin and resolution in causal arguments. In D. S. Weld and J. de Kleer, editors, *Readings in Qualitative Reasoning about Physical Systems*, pages 624–630. Kaufmann, San Mateo, CA, 1990.

[Kohonen, 1982] T. Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43:59–69, 1982.

[Kohonen, 1984] T. Kohonen. *Self-organization and associative memory.* Springer Verlag, 1984.

[Koller and Pfeffer, 1998] D. Koller and A. Pfeffer. Probabilistic frame-based systems. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 508–587, 1998.

[Konolige, 1994] K. Konolige. Using default and causal reasoning in diagnosis. *Annals of Mathematics and Artificial Intelligence*, 1994.

[Kononenko and Simec, 1995] I. Kononenko and E. Simec. Induction of decision trees using relieff. In *Proceedings of ISSEK Workshop on Mathematical and Statistical Methods in Artificial Intelligence*, pages 199–220, Udine, 1995.

[Kononenko, 1994] I. Kononenko. Estimating attributes: Analysis and extensions of relief. In *Proceedings of the 1994 European Conference on Machine Learning*, Springer-Verlag, 1994.

[Koton, 1989] P. Koton. Evaluating case-based problem solving. In *Proceedings of a Workshop on Case-Based Reasoning*, pages 173–175, Pensacola Beach, FL, 1989.

[Kowalski and Orlowska-Kowalska, 2003] T. Kowalski and Teresa Orlowska-Kowalska. Neural networks application for induction motor faults diagnosis. *Math. Comput. Simul.*, 63(3-5):435–448, 2003.

[Köpen-Seliger *et al.*, 2003] B. Köpen-Seliger, T. Marcu, M. Capobianco, S. Gentil, M. Albert, and S. Latzel. Magic: An integrated approach for diagnostic data management and operator support. In *Proceedings of the IFAC Symposium SAFEPROCESS*, pages 187–192, Washinton, USA, 2003.

[Kramer and Widmer, 2000] Stefan Kramer and Gerhard Widmer. Inducing classification and regression trees in first order logic. pages 140–156, 2000.

[Kuipers, 1986] B. J. Kuipers. Qualitative simulation. *Artificial Intelligence*, 29, 1986.

[Kuipers, 1990] B. Kuipers. Qualitative simulation. In D. S. Weld and J. de Kleer, editors, *Readings in Qualitative Reasoning about Physical Systems*, pages 236–260. Kaufmann, San Mateo, CA, 1990.

[Kulikowski and Weiss., 1982] C. A. Kulikowski and S. M. Weiss. Representation of expert knowledge for consultation: The casnet and expert project. *P. Szolovits, editor, Artificial Intelligence in Medicine, A.A.A.S Selected Symposium*, 51:21–55, 1982.

[Lachenbruch and Mickey, 1968] P. Lachenbruch and R. Mickey. Estimation of error rates in discriminant analysis. *Technometrics*, 10:1–11, 1968.

[Lanzola *et al.*, 1990] G. Lanzola, M. Stefanelli, G. Barosi, and L. Magnani. Neoanemia: a knowledge-based system emulating diagnostic reasoning. *Computers and Biomedical Research*, 23:560–582, 1990.

[Lee, 2005] Hong-Hee Lee. A study on decision tree for induction motor diagnostic system. In *2005 International Symposium on Electrical-Electronics Engineering-ISEE2005*, Ho Chi Minh City, VIETNAM, October 2005.

[Lennox *et al.*, 1998] B Lennox, P. Rutherford, G.A. Montague, and C. Haughin. Case study investigating the application of neural networks for process modeling and condition monitoring. *Computers and Chemical Engineering*, 22(11):1573–1579, 1998.

[Lim *et al.*, 1996] S. S. Lim, R. H. Lee, E.Ñ. Lim, and K. A. Ngoi. Multiple domain feature mapping utilizing case-base reasoning and blackboard technology. In *Proceedings of the Ninth International Conference on Industrial and Engineering Applications of Artificial Intelligent and Experts Systems (IEA/AIE - 96)*, Fukuoka, Japon, 1996.

[Lindenbaum *et al.*, 1999] M. Lindenbaum, S. Markovitch, and D. Rusakov. Selective sampling for nearest neighbor classifiers. *American Association for Artificial Intelligence*, 1999.

[Linguraru *et al.*, 2006] M.G. Linguraru, K. Marias, R. English, and J.M. Brady. A biologically inspired algorithm for microcalcification cluster detection. *Medical Image Analysis*, 2006.

[Liu and Setiono, 1995] H. Liu and R. Setiono. Chi2: Feature selection and discretization of numeric attributes. In *Proceedings of the Seventh IEEE International Conference on Tools with Artificial Intelligence*, 1995.

[Liu *et al.*, 2000] Xiang Qun Liu, Hong Yue Zhang, Jun Liu, and Jing Yang. Fault detection and diagnosis of permanent-magnet dc motor based on parameter estimation and neural network. *IEEE Transactions On Industrial Electronics*, 47(5), October 2000.

[Liu *et al.*, 2006] Shirong Liu, Qijiang Yu, and Jinshou Yu. A new on-line modeling approach to nonlinear dynamic systems. In *ISNN (2)*, pages 771–776, 2006.

[Lunze and Schiller, 1999] J. Lunze and F. Schiller. An example of fault diagnosis by means of probabilistic logic reasoning. *Control Engineering Practice*, 1999.

[Lunze, 1998] J. Lunze. Process supervisión by means of qualitative models. *Automation in Mining, Mineral and Metal Processing*, 1998.

[Malik and Struss, 1996] A. Malik and P. Struss. Diagnosis of dynamic systems does not necessarily require simulation. In *Workshop Notes of the Seventh International Workshop on Principles of Diagnosis DX-96*, Montreal, 1996.

[McCulloch and Pitts, 1943] W. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.

[McIlraith *et al.*, 1999] S. McIlraith, G. Biswass, D. Clancy, and V. Gupta. Towards diagnosing hybrid systems. In *Proceedings of the 10th International Workshop on Principles of Diagnosis*, 1999.

[Melle, 1980] W. J. Van Melle. *System Aids in Constructing Programs.* PhD thesis, University of Michigan, 1980.

[Milne and Travé-Massuyes, 1993] R. Milne and L. Travé-Massuyes. Real time model based diagnosis of gas turbine. In *Proceedings of the AIENG'93 International Conference*, Toulouse, 1993.

[Milne, 1992] R. Milne. On line diagnostic expert system for gas turbines. In *Proceedings of the 4th International Profiable Condition Monitoring Conference*, Stratford-Upon-Avon, 1992.

[Minsky and Papert, 1969] Marvin Minsky and Seymour Papert. *Perceptrons: An Introduction to Computational Geometry.* MIT Press, Cambridge, Mass., 1969.

[Moody and Darken, 1989] J. Moody and C. Darken. Fast learning in networks of locally-tuned processing units. *Neural Computation*, 1:281–294, 1989.

[Mosterman, 1997] P. Mosterman. *Hybrid dynamic systems: a hybrid bond graph modeling paradigm and its applications in diagnosis.* PhD thesis, Vanderbilt University, Nashville, Tennessee, USA, 1997.

[Muggleton, 1991] S. Muggleton. Inductive logic programming. *New Generation Computing*, 8(4):295–318, 1991.

[Murthy *et al.*, 1994] S.K. Murthy, S. Kasif, and S. Salzberg. A system for induction of obliques decision trees. *Journal or Artificial Intelligence Research*, 2:1–32, 1994.

[Nakatami *et al.*, 1996] Y.Ñakatami, M. Tsukiyama, and T. Wake. Plant fault diagnosis by integrating fault cases and rules. In *In Proceedings of the Ninth International Conference on Industrial and Engineering Applications of Artificial Intelligent and Experts Systems (IEA/AIE - 96)*, pages 95–102, Fukuoka, Japon, 1996.

[Ng, 1990] H. T. Ng. Model-based, multiple fault diagnosis of time-varying , continuous physical devices. In *Proceedings of the 6th IEEE Conference On AI Applications*, Santa Barbara, CA, 1990.

[Ng, 1991] H. T. Ng. Model-based, multiple fault diagnosis of dynamic, continuous physical devices. *IEEE Expert*, 6(6), 1991.

[Pacheco *et al.*, 2001] M. A. Pacheco, A. Arnanz, and J. R. Perán. Monitorización y detección de fallos en robots de soldadura. In *Actas de las I Jornadas de Diagnosis, Razonamiento Cualitativo y Sistemas Socioeconómicos*, Valladolid, 2001. Carlos Alonso y Juan Antonio Ortega, Editores.

[Panati and Drupé, 2000] A. Panati and D. T. Drupé. Stated based vs simulation-based diagnosis of dynamic system. In *Proceedings of the 14th European Conference on Artificial Intelligent*, 2000.

[Panati *et al.*, 2000] A. Panati, D. T. Drupé, and M. Formagnana. Causal simulation and diagnosis of dynamic systems. In *Proceedings of the 11th Workshop on Principles of Diagnosis*, Morelia, 2000.

[Patton *et al.*, 2000] R. J. Patton, P. M. Frank, and R.Ñ. Clark. *Issues of Fault Diganosis for Dynamic Systems.* Springer-Verlag, 2000.

[Perng *et al.*, 2000] Chang-Shing Perng, Haixun Wang, Sylvia R. Zhang, and D. Stott Parker. Landmarks: A new model for similarity-based pattern querying in time series databases. In *16th International Conference on Data Engineering (ICDE'2000)*, 2000.

[Piera, 1987] N. Piera. *Connectius de logiques no estandard com a operadors dàgregació en clasificació multivariable i reconeixement de formes.* PhD thesis, Universitat Politécnica de Cataluya, Barcelona, 1987.

[Poole *et al.*, 1987] D. Poole, R. Goebel, and R. Aleliunas. Theorist: A logical reasoning system for defaults and diagnosis. In N. Cercone and G. McCalla, editors, *The Knowledge Frontier*, pages 331–352. Springer, New York, 1987.

[Poole, 1994] D. Poole. Representing diagnosis knowledge. *Annals of Mathematics and Artificial Intelligence*, 1994.

[Pople, 1975] H. Pople. The dialog model of diagnostic logic an its use in internal medicine. In *Proceedings of the 4th international joint conference on artificial intelligent*, Tbilisi, USSR, 1975.

[Porter and Bareiss, 1987] B. Porter and E. Bareiss. Protos: An exemplar-based learning apprentice. In *Proceedings of the Fourth International Workshop on Machine Learning*, San Mateo, CA, 1987. Morgan Kaufmann.

[Pous *et al.*, 2003] Carles Pous, Joan Colomer, Joaquím Meléndez, and Josep Lluís de la Rosa. Case base management for analog circuit diagnosis improvement. In *ICCBR*, pages 437–451, 2003.

[Power and Bahri, 2004] Y. Power and A.P. Bahri. A two-step supervisory fault diagnosis framework. *Computers and Chemical Engineering*, (28):2131–2140, 2004.

[Price, 1999] C. Price. *Computer-based diagnositic systems.* Springer Verlag, 1999.

[Pulido *et al.*, 2001] B. Pulido, C. Alonso, and F. Acebes. Consistency-based diagnosis of dynamic systems using quantitative models and off-line dependency-recording. In *Proceedings of the 12th International Workshop on Principles of Diagnosis*, Sansicario, 2001.

[Quinlan, 1979] J.R. Quinlan. Discovering rules by induction from large collections of examples. In *Expert Systems in the Micho Electronic Age*, pages 168–201, Edinburgh, UK. Edinburgh University Press, 1979.

[Quinlan, 1993a] J. R. Quinlan. *C4.5: programs for machine learning.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[Quinlan, 1993b] J.R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann Publishers. San Mateo. California, 1993.

[R. *et al.*, 2002] R., Rafael M. Gasca, Carmelo Del Valle, and Miguel Toro. Max-csp approach for software diagnosis. In *IBERAMIA 2002: Proceedings of the 8th Ibero-American Conference on AI*, pages 172–181, London, UK, 2002. Springer-Verlag.

[Reiter, 1987] R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32, 1987.

[Riesbeck and Schank, 1989] C. K. Riesbeck and R. C. Schank. *Inside Case-Based Reasoning.* Erlbaum, Hillsdale, NJ, 1989.

[Rissanen, 1983] J. Rissanen. A universal prior for integers and estimation by minimum description length. *Annals of Statistics*, 11:416–431, 1983.

[Ritter *et al.*, 1975] G.L. Ritter, H.B. Woodruk, S.R. Lowdry, and T.L. Isenhour. An algorithm for a selective nearestneighbor decision rule. *IEEE Transactions on Information Theory*, 21(6):665–669, 1975.

[Rodriguez and Alonso, 2002] Juan J. Rodriguez and Carlos J. Alonso. Integración de un sistema de aprendizaje en la diagnosis basada en consistencia con modos de fallo. In *Actas de las IV jornadas ARCA. Sistemas Cualitativos y Diagnosis*, 2002.

[Rosenblatt, 1958] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

[Roverso, 2003] Davide Roverso. Fault diagnosis with the aladdin transient classifier. In *System Diagnosis and Prognosis: Security and Condition Monitoring*

Issues III. AeroSense2003, Aeroespace and Defense Sensing and Control Thecnologies Symposium, 2003.

[Sabin and Freuder, 1996] M. Sabin and C. Freuder. Automated construction of constraints based diagnosticians. In *Proceedings of the DX96*, 1996.

[Sabin *et al.*, 1995] D. Sabin, M. Sabin, R. Russell, and E. Freuder. A constraint-based approach to diagnosing software problems in computer networks. In *Proceedings of the First International Conference on Principles and Practice of Constraint Programming (CP'95)*, Cassis, France, 1995.

[Sainz *et al.*, 2002] M. A. Sainz, J. Armenglo, and J. Vehí. Fault detection and isolation of the three-tank system using the modal interval analysis. *Journal of Process Control*, 2002.

[Sainz Palmero *et al.*, 2005] G.I. Sainz Palmero, J. Juez Santamaria, E.J. Moya de la Torre, and J.R. Peran Gonzalez. Fault detection and fuzzy rule extraction in ac motors by a neuro-fuzzy art-based system. *Engineering Applications of Artificial Intelligence*, 18(7):867–874, 10 2005.

[Saludes *et al.*, 2001] S. Saludes, A. Vargas, and J. R. Perán. Aplicación de la red neuronal som para la detección de fallos desconocidos en un grupo hidroeléctrico. In *Actas de las I Jornadas de Diagnosis, Razonamiento Cualitativo y Sistemas Socioeconómicos*. Carlos Alonso y Juan Antonio Ortega, Editores, 2001.

[Samanta and Al-Balushi, 2003] B. Samanta and K. R. Al-Balushi. Artificial Neural Network Based Fault Diagnostics of Rolling Element Bearings Using Time-Domain Features. *Mechanical Systems and Signal Processing*, 17:317–328, March 2003.

[Saunders and Gammerman, 2000] C. Saunders and A. Gammerman. Application of support vector machines to fault diagnosis and automated repair. In *Proceedings of the Eleventh International Workshop on Principles of Diagnosis (DX-00)*, Morelia, Michoacen, Mexico, June 8-10 2000.

[Scarl, 1994] E. Scarl. Sensor placement for diagnosability. *Annals of Math and AI*, 11, 1994.

[Schoen *et al.*, 1995] R. R. Schoen, T. G. Habetler, F. Kamran, and R. G. Bartheld. Motor bearing damage detection current monitoring. *IEEE Trans. Ind. Applicat*, 31, Nov-Dec 1995.

[Seker *et al.*, 2003] Serhat Seker, Emine Ayaz, and Erdinc Turkcan. Elman's recurrent neural network applications to condition monitoring in nuclear power plant and rotating machinery. *Engineering Applications of Artificial Intelligence*, 16(7-8):647–656, 10 2003.

[Shortliffe *et al.*, 1981] E. H. Shortliffe, A. C. Scott, M. B. Bischoff, A. B. Campbell, W. Van Melle, and C. D. Jacobs. Oncocin: an expert system for oncology protocol management. In *Proceedings of the IJCA1-81*, 1981.

[Simón *et al.*, 2001] A. Simón, L. Alonso, and A. Antón. Sistema híbrido borroso para la ayuda del diagnóstico del glaucoma. In *Actas de las I Jornadas de Diagnosis, Razonamiento Cualitativo y Sistemas Socioeconómicos*, Valladilid, 2001. Carlos Alonso y Juan Antonio Ortega, Editores.

[Simoudis, 1992] E. Simoudis. Using case-based retrieval for customer technical support. *IEEE Expert: Intelligent Systems and Their Applications*, 1992.

[S.O.T. Ogaji and Prober, 2005] S. Sampath R. Singh S.O.T. Ogaji, L. Marinai and S.D. Prober. Gas-turbine fault diagnostics: a fuzzy-logic approach. *Applied Energy*, 82(1):81–99, September 2005.

[Sqalli and Freuder, 1998] M. H. Sqalli and E. C. Freuder. Diagnosing interoperability problems by enhancing constraint satisfaction with case-based reasoning. In *Proceedings of the 9th International Workshop on Principles of Diagnosis*, 1998.

[Srinivas, 1994] S. Srinivas. A probabilistic approach to hierarchical model-based diagnosis. In *Knowledge Systems Laboratory*. 1994.

[Stanfill and Waltz, 1986] C. Stanfill and D. Waltz. Toward memory-based reasoning. *Communications of the ACM*, 29:1213–1228, 1986.

[Steyer, 1991] J. P. Steyer. *Sur Une approche Qualitative des Systemes Physiques.Aide en temps reel a la conduite des procedes fermentaires.* PhD thesis, Paul Sabatier University, Toulouse, 1991.

[Stone, 1974] M. Stone. *Cross-validatory choice and assessment of statistical predictions*. C. J. Roy. Statist. Soc, 1974.

[Struss and Heller, 1999] P. Struss and U. Heller. Model-based support for water treatment. qualitative and model based reasoning for complex systems and their control. In *Proceedings of the Workshop KRR-4 at the 16th International Joint Conference on Artificial Intelligent*, 1999.

[Struss *et al.*, 1997] P. Struss, M. Sachenbacher, and F. Dummert. Diagnosing dynamic system with (almost) no observation. a case study in off-board diagnosis of the hydraulic of an anti-lock braking system. In *Proceedings of the 11th International Workshop on Qualitative Reasoning*, 1997.

[Struss, 1997] P. Struss. Fundamentals of model-based diagnosis of dynamic systems. In *Proceedings of the IJCAI'97*, 1997.

[Subramanian and Mooney, 1996] S. Subramanian and R. J. Mooney. Qualitative multiple-fault diagnosis of continuous dynamic systems using behavioral mode. In *Proceedings of the Thirteenth National Conference on Artifial Intelligence*, 1996.

[Sycara *et al.*, 1991] K. Sycara, D.Ñavin Chandra, R. Guttal, J. Koning, and S.Ñarasimhan. Cadet: a case-based synthesis tool for engineering design. *International Journal of Expert Systems*, 4(2):157–188, 1991.

[Tay and Shen, 2003] Francis E. H. Tay and Lixiang Shen. Fault diagnosis based on rough set theory. *Engineering Applications of Artificial Intelligence*, 16(1):19–43, Feb 2003.

[Tezafestas, 1987] S. G. Tezafestas. A look at the knowledge-based approach to system faults diagnosis and supervisory control. In Singh Tzafestas and editors schmidt, editors, *System Fault Diagnostics, Reliability and Related Knowledge-base Approaches,volume 2*, pages 3–16. D. Reidel Publishing Co., 1987.

[Thomas, 2002] P. Thomas. Fault detection and diagnosis in engineering systems. *Control Engineering Practice*, 10(3):1037–1038, 2002.

[Thornton, 1997] C. Thornton. Separability is a learner's best friend. In J. A. Bullinaria, D. W. Glasspool, and G. Houghton (Eds.), editors, *Proceedings of the Fourth Neural Computation and Psychology Workshop: Connectionist Representations*, pages 40–47, Berlin, 1997. Springer-Verlag.

[Tomek, 1976] I. Tomek. An experiment with the edited nearest-neighbor rule. *IEEE Transactions on Systems, Man and Cybernetics*, 6(6):448–452, 1976.

[Travé-Massuyes and Milne, 1998] L. Travé-Massuyes and R. Milne. Gaps between research and industry related to model-based and qualitative reasoning. In *Proceedings of the Model-Based Reasoning Workshop held at the thirteenth Bienal European Conference on Artificial Intelligent (ECAI)*, 1998.

[Travé-Massuyes *et al.*, 1993] L. Travé-Massuyes, K. Bousson, and J. M. Evrard. Non-causal vs causal qualitative modelling and simulation. *Intelligent Systems Engineering Journal*, 1993.

[Travé-Massuyes *et al.*, 2001] L. Travé-Massuyes, T. Escobet, and R. Milne. Model-based diagnosability and sensor placement. application to a frame 6 gasturbine subsystem. In *Proceedings of the 12th International Workshop on Principles of Diagnosis*, San Sicario, 2001.

[TravéMassuyes, 1992] L. TravéMassuyes. Qualitative reasoning for dynamical system simulation. In Sign M. (ed.) 2nd supplementary volume, editor, *Systems and Control Encyclopedia*. Pergamon Press, 1992.

[Turksen, 1998] I. B. Turksen. Fuzzy data mining and expert system development. In *Proceedings IEEE International Conference on Systems, Man, and Cybernetics*, pages 2057–2061, 1998.

[Utgoff and Brodley, 1990] P.E. Utgoff and C.E. Brodley. An incremental method for finding multivariate splits for decision trees. In *Proceedings of the Seventh International Conference on Machine Learning*, pages 58–65, Morgan Kaufmann, Los Altos, CA, 1990.

[Van de Merckt, 1992] T. Van de Merckt. Nfdt: A system that learns flexible concepts based on decision trees for numerical attributes. In *Proceedings of the 9th International Workshop on Machine Learning*, pages 322–331, 1992.

[Van de Merckt, 1993] T. Van de Merckt. Decision trees in numerical attribute spaces. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pages 1016–1021, Morgan Kaufmann, San Mateo, CA, 1993.

[Vapnik, 1998] VladimirÑ. Vapnik. *Statistical Learning Theory.* Wiley-Interscience, September 1998.

[Vemuri and Polycarpou, 2004] Arun T. Vemuri and Marios M. Polycarpou. A methodology for fault diagnosis in robotic systems using neural networks. *Robotica*, 22(4):419–438, 2004.

[Vinson *et al.*, 1992] J. M. Vinson, S. D. Grantham, and L. H. Ungar. Automatic rebuilding of qualitative models for diagnosis. *IEEE Intelligent Systems*, 1992.

[Vlahou *et al.*, 2003] A. Vlahou, J.O. Schorge, B.W. Gregory, and R.L. Coleman. Diagnosis of ovarian cancer using decision tree classification of mass spectral data. *Journal of Biomedicine and Biotechnology*, 2003(5):308–314, 2003.

[Wang *et al.*, 1997] W. Wang, J. Yang, and R. Muntz. Sting: A statistical information grid approach to spatial data mining. In *Proceedings of the 23rd International Conference on Very Large Data Bases*, pages 186–195, Athens, Greece. Morgan Kaufmann Publishers, 1997.

[Washio and Motoda, 2003] T. Washio and H. Motoda. State of the art of graph-based data mining. *SIGKDD Explorations*, 5(1):59–68, 2003.

[Widodo and Yang, 2007] Achmad Widodo and Bo-Suk Yang. Application of non-linear feature extraction and support vector machines for fault diagnosis of induction motors. *Expert Syst. Appl.*, 33(1):241–250, 2007.

[Wilson and Martínez, 1997] D.R. Wilson and T.R. Martínez. Improved heterogeneus distance functions. *Journal of Artificial Intelligence Research*, 6(1):1–34, 1997.

[Wilson, 1972] D. Wilson. Asymtotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man and Cybernetics*, 2(3):408–421, 1972.

[Wolpert, 1992] David H. Wolpert. Stacked generalization. *Neural Network*, 5(2):241–259, 1992.

[Wotawa, 1996] Franz Wotawa. *Applying Model-based Diagnosis to Software Debugging of Concurrent and Sequential Imperative Programming Languages*. PhD thesis, Technische Universität Wien, 1996.

[Wörn *et al.*, 2004] H. Wörn, T. Längle, M. Albert, A. Kazi, Attilio Brighenti, Santiago Revuelta Seijo, Christopher Senior, Miguel Angel Sanz Bobi, and Jose Villar Collado. Diamond: distributed multi-agent architecture for monitoring and diagnosis. *Production Planning and Control*, 15(2):189–200, March 2004.

[Yahui *et al.*, 2007] Su Yahui, Shen Zhao, Qian Honggang, Ma Honggang, Ji Jiafu, Ma Hong, Ma Longhua, Zhang Weihua, Meng Ling, Li Zhenfu, Wu Jian, Jin Genglin, Zhang Jianzhi, and Shou Chengchao. Diagnosis of gastric cancer using decision tree classification of mass spectral data. *Cancer Science*, 98(1):37–43, 2007.

[Yang *et al.*, 2004] B. S. Yang, T. Han, and J. L. An, art-kohonen neural network for fault diagnosis of rotating machinery. *Mechanical Systems and Signal Processing*, 18(3):645–657, May 2004.

[Yu *et al.*, 2003] Qian Yu, Xiuxi Li, Yanrong Jiang, and Yanqin Wen. An expert system for real-time fault diagnosis of complex chemical processes. *Expert Syst. Appl.*, 24(4):425–432, 2003.

[Yuan and Chu, 2006] Sheng-Fa Yuan and Fu-Lei Chu. Support vector machines-based fault diagnosis for turbo-pump rotor. *Mechanical Systems and Signal Processing*, 20(4):939–952, May 2006.

[Yvan *et al.*, 2005] Yvan, Thomas Zimmer, and André Ivanov. An analog circuit fault characterization methodology. *J. Electron. Test.*, 21(2):127–134, 2005.

[Zadeh, 1965] L. A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.

[Zhao *et al.*, 2005] Gang Zhao, DongXiang Jiang, Kai Li, and JinHui KeyDiao. Data mining for fault diagnosis and machine learning for rotating machinery. *Engineering Materials*, 293-294:175–182, 2005.