UNIVERSIDAD AUTÓNOMA METROPOLITANA

Minería Sobre Grandes
Cantidades de Datos

Para obtener el grado de
MAESTRO EN CIENCIAS
(Ciencias y Tecnologías de la Información)

PRESENTA:

Lic. Benjamín Moreno Montiel

Asesor de la Tesis

Dr. René Mac Kinney Romero

Sinodales

Presidente: Dr. Eduardo Morales Manzanares

Secretario: Dr. René Mac Kinney Romero

Vocal: Dr. John Goddard Close

Vocal: M. en C. Alma Edith Martínez Licona

México D.F
Noviembre del 2009

Agradecimientos

Al Consejo Nacional de Ciencia y Tecnología, CONACyT, por la beca otorgada para estudios de posgrado.

A la Universidad Autónoma Metropolitana - Unidad Iztapalapa y a todos los Académicos que han sido parte importante de mi formación.

Un agradecimiento muy especial a mi Asesor el Dr. René Mac Kinney Romero, por todos los conocimientos que comparte conmigo y por la dedicación y el gran apoyo brindado en la elaboración de la presente Tesis.

Dedicado a...

Esta tesis está dedicada con todo mi cariño a mi madre, que con su apoyo, comprensión y enseñanzas ha sido pieza fundamental para lograr siempre mis metas académicas y personales..

También esta dedicada a Carlos, Eli, Gamo, Hugo, Noemi, y Victor por su apoyo incondicional y gran compresión.

Contenido

Lista de Figuras

Lista de Tablas

Capítulo 1

Introducción

Una de las áreas que ha tenido más avances en la ultima década, ha sido la Minería de Datos, debido principalmente al incremento en el tamaño de las Bases de Datos (BD), teniendo como resultado una falta de conocimiento de la información que se encuentra presente en ellas. Si tenemos una gran cantidad de datos y de estos no sabemos que información útil existe, es igual o peor que no tener ninguna información de ellos. Esta es la razón por la que el objetivo principal de investigaciones sobre Minería de Datos, presentan la forma de realizar dicho descubrimiento del conocimiento inmerso dentro de grandes BD.

Un ejemplo de estas BD, son las provistas por The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in DB 2007[10] (ECML PKDD 2007). ECML PKDD es uno de los congresos más importantes en la área de Aprendizaje Computacional (*Machine Learning*) y Minería de Datos, realizado en Europa cada año. Se han llevado a cabo 19 EMCL y 12 conferencias de PKDD y en cada conferencia es provista una BD como un reto para extraer conocimiento de ella.

Minería de Datos es la exploración y análisis para la identificación no trivial de patrones (conocimiento) dentro de grandes cantidades de datos, que puedan ser válidos, novedosos, potencialmente útiles y entendibles. Los resultados que obtengamos al aplicar Minería de Datos deben de ser interesantes, siempre y cuando sean comprensibles (por seres humanos), sean válidos con cierto grado de certeza, sean potencialmente útiles y deben de ser novedosos para validar una hipótesis planteada, una vez que se tiene alguno de estos resultados se pueden evaluar de manera *objetivamente* (criterios estadísticos) ó *subjetivamente* (perspectiva del usuario). Cuando se lleva acabo una

tarea de Minería de Datos, se puede describir en términos de:

- **Datos relevantes:** Dentro de todos los datos que vamos a tener, debemos de seleccionar aquellos que sean más importantes para analizar.
- **Conocimiento previo (*Background knowledge*):** Se debe de tener un conocimiento parcial para que se pueda guiar el proceso de forma correcta.
- **Tipos de conocimiento:** Siempre que se plantea el problema de encontrar un conocimiento sobre un conjunto de datos, se deben de establecer qué es lo que se desea obtener, usualmente a esto se le conoce como la *problemática* asociado a la BD.
- **Medidas de rendimiento:** Se deben de plantear estas medidas estadísticas para evaluar los resultados obtenidos.
- **Técnicas de representación:** Siempre se debe tener establecida cuál va a ser la forma de representar los datos que estemos manejando.

La Minería de Datos es una fase importante para poder llevar a cabo el proceso de Extracción de conocimiento en Bases de Datos (*Knowledge Discovery in Databases KDD*), la cual puede verse como una área de investigación multidisciplinaria, ya que diversas áreas de investigación aportan un beneficio para que en conjunto se pueda llevar acabo la Minería de Datos. A continuación se listarán dichas áreas:

1. **Bases de Datos:** Nos permite gestionar grandes cantidades de datos que nos permitan almacenar y posteriormente acceder a los datos de forma rápida y estructurada.
2. **Estadística:** Cada vez que se la aplica Minería de Datos los resultados se evalúan utilizando medidas estadísticas, para poder obtener un resumen de los datos utilizados.
3. **Inteligencia Artificial:** De esta importante área de investigación en las Ciencias de Computación, radican métodos de aprendizaje que nos van a permitir llevar a cabo una de las metas principales en Minería de Datos llamada clasificación.

4. **Visualización:** Con esta disciplina podemos presentar el conocimiento y los datos usados, de una manera más visual y comprensible.

El nombre de Minería de Datos viene por la similitud con lo realizado cuando se construye una Mina, para poder extraer algún mineral o material que produzca un beneficio económico. Al seleccionar una montaña se desconoce cuanto material esta presente, por lo que desconocemos si será un proyecto exitoso o no. Una vez que se decide explorar alguna montaña, se construye una mina, con la cual podremos explorar la montaña hasta obtener una meta, ya sea en cuestión de beneficios económicos o de una cierta cantidad del material extraído. En esta mina se empiezan los trabajos de exploración y excavación de dicho material por medio de los mineros, realizando posteriormente un proceso de transporte que nos sirve para sacar lo extraído en la mina y hacer la separación de lo que realmente es útil. Al final cuando el objetivo del proyecto es cubierto se dice que la mina es cerrada y se garantiza que no se afecta a la naturaleza (en la mayoría de los casos). Después de realizar este cierre se establecen los beneficios alcanzados y se saca una estadística de lo que se obtuvo durante el periodo en el que la mina se encontró vigente.

De forma similar podemos ver a la Minería de Datos, en este caso la montaña en cuestión es de datos, se lleva un orden para aplicar Minería de Datos mediante el proceso de KDD, los mineros en este caso son los clasificadores, utilizados para poder particionar los datos de acuerdo a las clases que presenten en un caso de clasificación, posteriormente se tiene que seleccionar cuales van a ser nuestras medidas de rendimiento, para evaluar que tan efectivo fue utilizar este proceso, y finalmente debemos de presentar el conocimiento que se obtuvo al utilizar dicha BD.

Existen diferentes congresos como EMCL PKDD, KDD *(ACM SIGKDD International Conference on KDD)*, ICDM *(IEEE International Conference on Data Mining)*, SDM *(SIAM Data Mining Conference)*, entre otros más, en donde se exponen aplicaciones de vanguardia con BD del mundo real, haciendo uso de Minería de Datos, a continuación mostraremos algunas aplicaciones donde puede ser utilizadas con Minería de Datos:

- **En la Universidad:** Dentro de la UAM se realizan dos estudios sobre los egresados de las diferentes carreras impartidas en los 4 planteles, Azcapozalco, Cuajimalca, Iztapalapa y Xochimilco, llamados *Estudio de Seguimiento de Egresados Generaciones 1998 y 2003*[34]

y *Estudio de Empleadores y Tendencias del Mercado Laboral Egresados Generaciones 1997 y 2002*[35], en donde se puede ver una aplicación de Minería de Datos, ya que se contaba con encuestas realizadas a un número considerable de egresados, por lo que se generó una BD grande, de la cual se calcularon estadísticas para saber la información referente a los egresados, como cuantos de ellos salieron en un periodo de 4 años y cuantos salieron en un periodo mayor a este, que porcentaje se encontraba trabajando en la industria así como el porcentaje de los que siguieron sus estudios de posgrado. Este tipo de información se puede calcular por medio de medidas estadísticas que nos arrojen resultados parciales de la BD, la aplicación de Minería de Datos va más haya de estas medidas, al aplicarla podemos encontrar alguna correlación entre diferentes atributos, formando combinaciones que predigan el tiempo en el que un determinado alumno finalice sus estudios. Otra información que podemos obtener mediante la aplicación de Minería de Datos es saber si en un conjunto de alumnos dentro de una zona geográfica dada, las escuelas de nivel medio superior contribuyeron en el rendimiento que presentaron a lo largo de su estancia en la UAM. Esto puede ser de ayuda para tener otra predicción del rendimiento que van a a tener nuevas generaciones de acuerdo a la zona geográfica a la que pertenezcan. Estas son solo dos de un gran número de aplicaciones que se le puede dar a la Minería de Datos, específicamente para este ejemplo.

- **En Empresas:** Por el año de 2001, la industria financiera mundial tuvo una perdida de cerca de 2,000 millones de dólares, en fraudes que llevaron a cabo con tarjetas de crédito. Es por está razón que se desarrolló un sistema inteligente llamado *Falcon Fraud Manager* de Fair Isaac [12], capaz de detectar y mitigar fraudes mediante la exploración de las BD de propietarios de tarjetas de crédito, transacciones realizadas por los mismos propietarios y datos financieros. En un principio su uso se le dió directamente al servicio financiero Norteamericano, para la detección de fraudes en tarjetas de crédito, pero en la actualidad se encuentra en diversas empresas mundiales no solo para detectar los fraudes en este tipo de tarjetas, también se incorporaron datos de tarjetas comerciales, de combustible y de crédito. Este sistema ha permitido ahorrar más de 700 millones de dólares por año a la industria financiera y garantiza la protección de más de 450 millones de pagos con tarjeta de crédito en todo el

mundo (cerca del 75 % de todas las transacciones con tarjeta de credito). La aplicación que tiene Minería de Datos para este ejemplo, es detectar en un tiempo real clientes fraudulentos que tengan algun tipo de tarjeta de pago, mediante patrones específicos, como depósitos y retiros de la tarjeta o pagos que se tengan con la misma y así poder ofrecer a los grupos financieros un mayor grado de seguridad a sus servicios de creditos, evitando de esta manera fraudes millonarios. Este es otro ejemplo en donde la Minería de Datos se encuentra presente apoyando a la seguridad de un gran número de países a nivel mundial.

Una de las tareas principales de KDD y Minería de Datos, es la *clasificación*. La clasificación es usada para predecir el tipo, o como usualmente se le nombra la clase de cada ejemplo presente en los datos y es realizada mediante diversos tipos de *clasificadores*. De acuerdo a J. Ross Quinlan[38], los siguientes tipos de clasificadores son los más utilizados para efectuar la clasificación de los datos.

- Tablas de Decisión.
- Reglas de Decisión.
- Clasificadores Basados en Casos.
- Redes Neuronales.
- Clasificadores Bayesianos.
- Clasificadores basados en Acoplamientos.

Nuestro trabajo estará enfocado a revisar detalladamente los clasificadores basados en acoplamientos, en los cuales se combina un conjunto de diferentes clasificadores para mejorar la exactitud. Bauer et. al.[1], Schapire[32], Koltchinskii[21] y otros más, consideran la construcción de estos clasificadores usando diferentes modelos de Árboles de Decisión. Estos clasificadores serán revisados más a detalle en el Capítulo 4:

El tamaño de la BD es importante para saber si es adecuado aplicar Minería de Datos, es por eso que se escogió la clasificación de las BD de acuerdo a “The HuberWegman Taxonomy of Data Set Sizes”[18], que se muestra en la Tabla 1.1.

Tabla 1.1: Clasificación de las BD.

Caso	*Clasificación*	*Tamaño de la BD*
1	Diminutas (*Tiny*)	$\leq$ 100 Bytes
2	Pequeñas (*Small*)	$\leq$ 10 Kilobytes
3	Medianas (*Medium*)	$\leq$ 1 Megabytes
4	Largas (*Large*)	$\leq$ 100 Megabytes
5	Enormes (*Huge*)	$\leq$ 10 Gigabytes
6	Masivas (*Massive*)	$\leq$ 1 Terabytes

La principal BD usada para este trabajo es una BD del mundo real, la cual cae dentro de la clasificación cuatro (Large) de la Tabla 1.1, con la que se mostrarán los resultados obtenidos al realizar una serie de pruebas con diferentes tipos de clasificadores usados y con nuestra propuesta. Al terminar de mostrar los resultados obtenidos con esta BD, mostraremos otros resultados obtenidos con algunas BD estándar del repositorio de la Universidad de California en Irvine[4]

El objetivo de este trabajo es mostrar la construcción de un sistema de mezcla de expertos, que es un tipo de clasificadores basados en acoplamientos. Al realizar un sistema de mezcla de expertos presentamos dos problemas en particular. El primer problema es decidir cuales clasificadores vamos a utilizar y si estos van a presentar buenos resultados al realizar una mezcla de expertos. El segundo problema que se presenta es la forma de como asignar los pesos a cada clasificador utilizado para poder implementar el criterio de votación ponderada y obtener así la clasificación de los datos. En este trabajo mostraremos un sistema de mezcla de expertos construido con diferentes tipos de clasificadores usando, Bayes Ingenuo, *K*-Medias, *k*-Vecinos más Cercanos, C4.5, Decision Tables y ADTree. usando un criterio de votación ponderada aplicando un algoritmo genético para encontrar la ponderación de los pesos asignados a cada clasificador. Este algoritmo lleva el nombre de Mezcla Genética de Expertos Ponderada (*Weighted Genetic Mixture of Experts* - WGME) y la idea general consiste en combinar un conjunto de clasificadores considerados, con un criterio de votación ponderada (cada clasificador tiene un peso diferente, asignado mediante el algoritmo genético), con el objetivo de encontrar una mejor clasificación que la que se obtiene de manera

individual con cada clasificador considerado. Se realizó la clasificación de diferentes conjuntos de prueba para medir el rendimiento de este sistema y se comparo con una serie de pruebas realizados sobre diferentes herramientas para aplicar Minería de Datos en los que se obtuvo un mejor rendimiento con el sistema propuesto.

Este trabajo se encuentra organizado de la siguiente forma. En el Capítulo 2 mencionaremos algunos conceptos importantes de Aprendizaje Computacional y describiremos a detalle cada uno de los métodos de clasificación considerados para la construcción del WGME. En el Capítulo 3 se revisará el proceso de extracción de conocimiento en Bases de Datos, explicando cada una de las fases que nos permiten encontrar el conocimiento en BD, veremos como utilizamos este proceso para la BD utilizada, en donde describiremos como se utilizó cada fase, sobre esta BD. En el Capítulo 4 se describirá a detalle nuestra propuesta, revisando a detalle los clasificadores basados en acoplamientos, que son la base para la construcción de nuestra propuesta, una vez revisados este tipo de clasificadores se explicara cada una de las partes que conforman nuestro Algoritmo de Votación Ponderada Multiclasificador, así como la aplicación que fue desarrollada en Matlab 7.7 (2008b). En el Capítulo 5 se describirán a detalle cuales fueron las pruebas que se efectuaron con esta BD en donde se utilizaron diferentes herramientas que nos permiten realizar Minería de Datos, reportadas en la literatura, presentando la comparación de los resultados obtenidos contra nuestra propuesta. Finalmente en el Capítulo 6 se daremos algunas conclusiones y cual es el trabajo a futuro que se podría realizar.

Capítulo 2

Aprendizaje Computacional

2.1. Introducción

La mayoría de las técnicas tradicionales dentro de la Inteligencia Artificial, tienen por objetivo desarrollar aplicaciones para dar una solución a problemas que presentan un cierto de grado de complejidad, en los que a veces la solución óptima no es factible de encontrar, e incluso desconocida, por lo que estas aplicaciones nos ofrecen soluciones parciales que suelen dar buenos resultados. Para construir dichas aplicaciones, los programadores deberán incluir conocimiento del dominio del problema para facilitar su resolución, logrando con esto la reducción del costo computacional que implica el uso de técnicas convencionales.

Con frecuencia estas aplicaciones suelen ser programadas para dar solución a un problema en específico, sin embargo al momento que se quiere generalizar el uso de estas para problemas similares, se necesita volver a programar y agregar conocimiento nuevo para resolver estos problemas. Vemos entonces que una de sus principales limitaciones radican en que no podrán resolver problemas para los que no hayan sido programadas. Es aquí donde el conocimiento que vayamos integrando aumenta el uso que le podemos dar a estas aplicaciones.

Una aplicación que sea capaz de adaptarse y poder integrar nuevo conocimiento, a medida que se usa, de manera automática se considera más cercana a ser realmente una aplicación inteligente, ya que podrá resolver nuevos problemas que nosotros le proporcionemos sin esperar una reprogramación y un conocimiento nuevo que le ayude a atacar estos problemas. El área de investigación llamada Aprendizaje Computacional, dentro de la Inteligencia Artificial, se encarga de desarrollar

técnicas para construir aplicaciones en las que se tenga una adaptación de conocimiento de manera automática.

La idea del Aprendizaje Computacional es buscar métodos que logren aumentar las capacidades de las aplicaciones habituales (sistemas basados en el conocimiento, tratamiento del lenguaje natural, robótica, etc.) de manera que puedan ser mas flexibles y eficaces. Hay muchas utilidades que se le pueden dar al aprendizaje computacional para la construcción de programas de inteligencia artificial, a continuación se muestran tres usos[2]:

- **Tareas difíciles de programar:** Es muy frecuente que nos encontremos con tareas excesivamente complejas en las que construir un programa para resolverlas resulta muy difícil. Pongamos el ejemplo en el que se pretende desarrollar un sistema de visión que sea capaz de reconocer un conjunto de rostros, sería muy complicado desarrollar un programa a mano que resuelva este reconocimiento. El aprendizaje computacional nos permite construir un modelo de clasificación a partir de un conjunto de ejemplos, para realizar la tarea del reconocimiento y de esta manera hacer la resolución del problema más sencilla.

- **Aplicaciones auto adaptables:** Hay ciertas aplicaciones que tienen un mejor rendimiento si son capaces de adaptarse a las circunstancias e ir aprendiendo a lo largo de su ejecución. Podemos tener por ejemplo aplicaciones de interfaces que sean adaptables al uso contínuo del usuario y poder ofrecer un mejor servicio. Otro ejemplo de aplicaciones auto adaptables son los gestores de correo electrónico, que son capaces de aprender a distinguir entre el correo no deseado y el correo normal, a medida que se utilizan.

- **Minería de Datos/Descubrimiento de conocimiento:** El aprendizaje puede servir para ayudar a analizar información, extrayendo de manera automática conocimiento a partir de conjuntos de ejemplos (usualmente millones) y descubriendo patrones complejos.

El aprendizaje computacional se divide en cuatro tipos de aprendizaje:

- **Aprendizaje Inductivo:** En este tipo de aprendizaje se crean modelos de conceptos o descriptores que posean características comunes de los ejemplos de entrenamiento.

- **Aprendizaje analítico o deductivo:** Es aplicada la deducción para obtener descriptores de un ejemplo, de un concepto y su descripción.

- **Aprendizaje genético:** Son aplicados algoritmos inspirados en la Teoría de la evolución para encontrar descriptores generales de los ejemplos.

- **Aprendizaje conexionista:** Se desarrollan descriptores generales mediante el uso de las capacidades de adaptación de redes neuronales artificiales. Una red neuronal esta compuesta de elementos simples interconectados que poseen algún estado. Tras un proceso de entrenamiento, el estado en el que quedan las neuronas de la red representa el concepto aprendido.

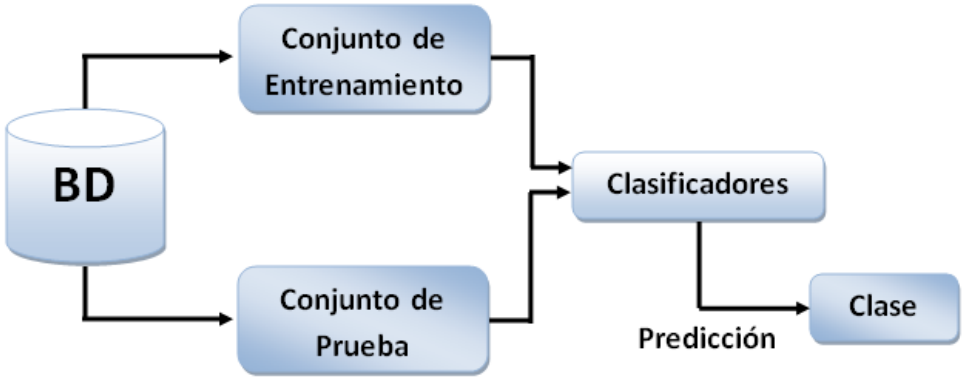
En este trabajo prestaremos mas atención al aprendizaje inductivo, en donde podemos distinguir al aprendizaje supervisado y al no supervisado:

- **aprendizaje supervisado:** En este tipo de aprendizaje se tiene un conjunto de ejemplos, los cuales tienen asociados un atributo llamado la clase de cada ejemplo, tendremos entonces algún mecanismo de entrenamiento en base a este conjunto, para permitirnos distinguir las clases sobre ejemplos de prueba que proporcionemos.

- **aprendizaje no supervisado:** En este tipo de aprendizaje solo se cuenta con un conjunto de ejemplos que no tiene asociado ninguna clase, por lo que se debe de encontrar una manera de agruparlos. Para los métodos de aprendizaje no supervisado se utiliza el concepto de similaridad/disimilaridad de los ejemplos, para poder construir grupos en los que ejemplos similares estén justos en un grupo y separados de otros ejemplos menos similares.

Lo que se pretende revisar a lo largo de este trabajo, son los métodos de clasificación más utilizados para estos dos tipos de aprendizaje. En la Figura 2.1 se muestra un esquema básico de un clasificador en el aprendizaje computacional, que muestra como de una BD se seleccionan dos conjuntos, uno de entrenamiento para aplicar una fase de entrenamiento a cada clasificador utilizado, para que de esta forma se pueda pasar a clasificar el conjunto de prueba del cual desconocemos la clase de cada ejemplo de prueba.

Figura 2.1: Esquema de un clasificador en aprendizaje computacional



Como primer punto se revisa una forma de clasificar a un conjunto de ejemplos mediante la construcción de Árboles de Decisión, usando los algoritmo *ID3 (Iterative Dichotomiser 3)* y *C4.5* [29, 28] que permite la construcción de dichos árboles. La construcción de árboles de decisión nos permite representar conocimiento a partir de un conjunto de ejemplos, con esto podemos realizar una generalización de ellos y hallar una forma de clasificar los ejemplos dados. Revisaremos un ejemplo práctico de como se construyen estos árboles y terminaremos con algunas ventajas y desventajas de este método de clasificación.

Continuaremos con una área de investigación de Inteligencia Artificial llamada *Análisis de Clusters*, en donde existen dos tipos de métodos para poder clasificar a un conjunto de datos, estos son los *Métodos jerárquicos de clusters* y los *Métodos no jerárquicos de clusters*. La diferencia entre ambos radica en la selección de los k clusters en los que se pretende clasificar al conjunto de datos, mientras que en los método de clusters jerárquicos se va construyendo dichos clusters de acuerdo a la similitud que tengan los datos, en la mayoría de los métodos no jerárquicos se fija el valor k de clusters para que de acuerdo a este valor los datos queden clasificados. Revisaremos el funcionamiento de los dos métodos y plantearemos un algoritmo básico para ambos casos y un ejemplo práctico para ver qué tipo de resultados arroja cada uno de estos agrupadores. Finalizaremos con algunas ventajas y desventajas de estos dos métodos de agrupación.

El siguiente clasificador que revisaremos es el *Método de los k vecinos más cercanos* el cual

se ocupa de predecir o clasificar un nuevo dato basado en observaciones conocidas o pasadas. Este es la generalización del enfoque del vecino más cercano, en donde solo se toma al ejemplo que se encuentre más cercano al nuevo dato. Se verá la justificación del porque se utiliza el Método de los k vecinos más cercanos, ademas se planteará un algoritmo básico que nos permita realizar este método y concluiremos con unas las ventajas y desventajas del método.

Pasáremos a revisar varios enfoques de cómo podemos realizar una clasificación de un conjunto de datos de prueba haciendo uso del concepto de probabilidad *a priori*, probabilidad condicional y probabilidad *a posteriori*, las cuales ayudan a decidir a qué clase pertenece cada uno de estos datos de prueba. Se vera un ejemplo de un clasificador llamado *Bayes Ingenuo* y finalizaremos con las ventajas y desventajas de este clasificador.

Finalmente revisaremos un clasificador llamado *Tablas de Decisión*, que cae en el tipo de clasificadores por *Reglas de Decisión*. Con este tipo de clasificadores nosotros podemos plantear las posibles acciones que deben de tomar, para resolver un problema de clasificación, construyendo reglas en estructuras tipo tablas, en las que estén presentes diferentes combinaciones de valores para cada situación presente en el problema de clasificación, las causas y las acciones que se deben de tomar para cada una de estas combinaciones de valores y finalmente las acciones o efectos que se deben de realizar. Se verá un ejemplo práctico del funcionamiento de las Tablas de Decisión y cuales son las ventajas y desventajas de usar este tipo de clasificadores.

2.2. Árboles de Decisión

2.2.1. Introducción

Un árbol de decisión es una forma gráfica y analítica para poder llevar a cabo la clasificación de los datos utilizados, mediante diferentes caminos posibles. Cada una de los nodos del árbol representa los diferentes atributos presentes en los datos, las ramificaciones del árbol representan los caminos posibles a seguir, para predecir la clase de un nuevo ejemplo, en donde los nodos terminales u hojas establecen la clase a la que pertenece el ejemplo de prueba si se sigue por la ramificación en cuestión.

El lenguaje de descripción de los árboles de decisión corresponde a las fórmulas en FND (Forma Normal Disyuntiva)[2]. Consideramos el caso en el que tenemos 3 atributos, el atributo A, B y C, cada uno de ellos con dos valores, x_i y $\neg\ x_i\ \forall\ i \equiv$ 1,2,3 respectivamente, con esto de pueden construir 2^n combinaciones en FNC (Forma Normal Conjuntiva). Un ejemplo de estas se muestra a continuación:

$$(x_1 \wedge x_2 \wedge \neg x_3)$$

Cada una de las combinaciones en FNC describen una parte del árbol que estemos formando, por lo que tendríamos para el árbol disyuntivas de la siguiente forma:

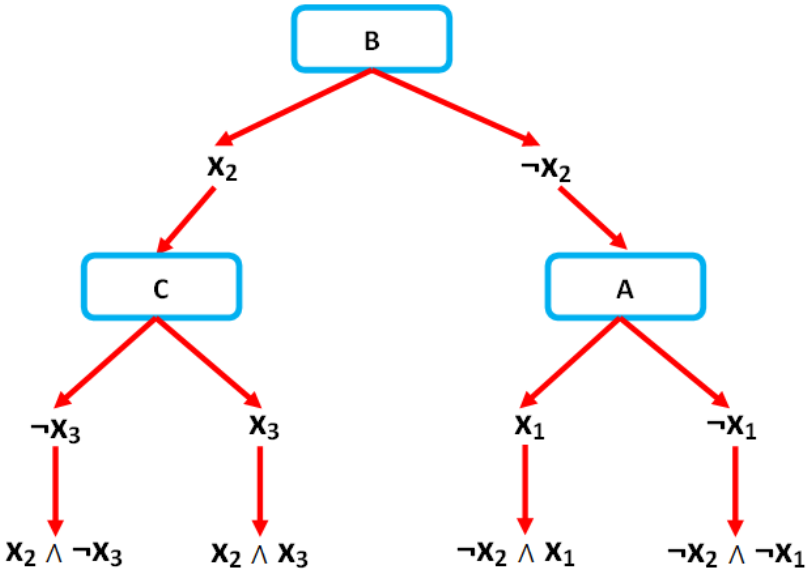
$$(x_2 \wedge \neg x_3) \vee (x_2 \wedge x_3) \vee (\neg x_2 \wedge x_1) \vee (\neg x_2 \wedge \neg x_1)$$

Las cuales son descriptores del árbol de inducción construido, entonces podríamos formar $O(2^{2^n})$ descripciones posibles en FND. El árbol correspondiente a los descriptores anteriores se muestra en la Figura 2.2.

Dado que el orden de los arboles o descriptores es muy grande, evidentemente no es posible explorar todos los descriptores para ver cual es el mas adecuado, por lo que se utilizan técnicas de búsqueda heurística para encontrar una forma más fácil y rápida de hacerlo.

La mayoría de los algoritmos de construcción de árboles de decisión se basa en la estrategia de

Figura 2.2: Aspecto visual de un árbol de decisión.



Ascenso a la Colina (*Hill Climbing*). Esta es una técnica utilizada en Inteligencia Artificial para encontrar los máximos o mínimos de una función mediante una búsqueda local. Estos algoritmos empiezan con un árbol vacío, después se va particionado en conjuntos de ejemplos, eligiendo en cada caso aquel atributo que mejor discrimina entre las clases, hasta que se completa el árbol. Para saber qué atributo es el mejor se utiliza una función heurística, la elección que hagamos es irrevocable, por lo que debemos asegurar que esta sea la más cercana a la óptima. La principal ventaja de usar este tipo de estrategias es que el costo computacional es bastante reducido.

2.3. Algoritmo ID3 (Iterative Dichotomiser 3)

El primer algoritmo que mostraremos es ID3 desarrollado por Quinlan[29] en 1983 el cual es considerado un algoritmo seminal, ya que de aquí se derivan muchos algoritmos para la construcción de árboles de decisión. Este algoritmo se basa en la teoría de la información, desarrollada en 1948 por Claude Elwood Shannon[31]. En particular se utiliza la noción de entropía descrita en la

teoría de la información, para ver que tan aleatorio se encuentra la distribución de un conjunto de ejemplos sobre las clases a las que pertenecen. Dentro de esta teoría de la información se estudia también cuales son los mecanismos de codificación de los mensajes y el costo asociado a su transmisión. Sea $M = \{m_1, m_2, \ldots, m_n\}$ un conjunto de mensajes, en donde para cada uno de ellos se tiene una probabilidad $P(m_i)$, entonces podemos definir la cantidad de información I contenida en un mensaje de M como[1]:

$$I(M) = \sum_{i=1}^{n} -P(m_i)\log_2(P(m_i))$$

Este valor nos dice que cantidad de bits de información son necesarios para codificar los diferentes mensajes de M. Dado un conjunto de mensajes, podemos obtener la mejor manera de codificarlos para que el coste de su transmisión sea mínimo. Además nos permite ir seleccionando al mejor atributo cuyo conocimiento aporte mayor información desde la perspectiva de clasificación, en cada uno de los niveles del árbol que se vaya construyendo. Podemos realizar una analogía con la codificación de mensajes tomando a las clases como los mensajes y la proporción de ejemplos de cada clase como su probabilidad. El objetivo de los árboles de decisión es construir el árbol de tamaño mínimo, que nos permita distinguir los ejemplos de cada clase. Cada atributo se deberá evaluar para decidir si se le incluye en el árbol. Un atributo será mejor cuanto más permita discriminar entre las diferentes clases como se había comentado anteriormente.

2.3.1. Funcionamiento de ID3

En este algoritmo se parte de un árbol vació y se va construyendo de manera recursiva, tomando en cada nodo aquel atributo que tiene el mayor grado de información, haciendo que sea menos la cantidad de información que falta por cubrir.

Cada vez que hagamos la elección de un atributo, debería hacer que los subconjuntos de ejemplos que genera el atributo sean mayoritariamente de una clase. Para medir esto necesitamos una

[1]Se multiplica por menos, debido a que como manejamos probabilidades que están definidas entre [0, 1], el $logaritmo$ de estas probabilidades nos da un número negativo, por lo que convertimos este valor en positivo para darle un sentido intuitivo.

medida de la cantidad de información que cubre un atributo (medida de *Entropía*, E). La primera cosa que debemos de formular es la *cantidad de información* que tiene cada clase la cual se puede hacer de la siguiente forma:

Dado un conjunto de ejemplos X clasificados en un conjunto de clases $C = \{c_1, c_2, \ldots c_n\}$, siendo $|c_i|$ la cardinalidad de la clase c_i y $|X|$ el número total de ejemplos, la cantidad de información vendrá expresada de la siguiente forma:

$$I(X, C) = -\sum_{c_i \in C} \frac{|c_i|}{|X|} \log_2 \left(\frac{|c_i|}{|X|} \right)$$

Teniendo la cantidad de información se puede formular la Entropía de la siguiente forma:

Para cada uno de los atributos A_i, siendo $\{v_{i1}, \ldots v_{in}\}$ el conjunto de posibles valores del atributo A_i y $|[Ai(C) = v_{ij}]|$ el número de ejemplos que tienen el valor v_{ij} en su atributo A_i, la función de entropía es expresada de la siguiente forma:

$$E(X, C, A_i) = -\sum_{v_{ij} \in A_i} \frac{|[A_i(C) = v_{ij}]|}{|X|} I([A_i(C) = v_{ij}], C)$$

Con el resultado de estas medidas se define la función de *ganancia de información* de la siguiente forma:

$$G(|X|, C, A_i) = I(X, C) - E(X, C, A_i)$$

Es por ello que se toma el atributo que maximice este valor para ir expandiendo el árbol de inducción. El pseudocódigo del algoritmo ID3 para la construcción de un árbol de decisión es el siguiente:

func *ID3*(X, C, A) $\equiv$ (X: Ejemplos, C: Clasificación, A: Atributos)

si todos los ejemplos son de la misma clase

entonces *ID3* $\leftarrow$ hoja con la clase.

otro

Calcular la función de cantidad de información de los ejemplos (I)

para cada atributo en A

Calcular la función de entropía (E) y la ganancia de información (G)

Escoger el atributo que maximiza (G) (sea a)

Eliminar a de la lista de atributos (A)

termina

para cada partición generada por los valores v_i del atributo a

$Arbol_i \leftarrow ID3$(ejemplos de X con $a = v_i$,

Clasificación de los ejemplos, Atributos restantes)

Generar árbol con $a = v_i$y $Arbol_i$

termina

ID3 $\leftarrow$ la unión de todos los árboles

.

2.3.2. Ejemplo de uso del Algoritmo ID3

Un ejemplo sencillo es el siguiente en el cual se decide que día es el aceptado para poder jugar un partido de golf [26]. El conjunto de ejemplos se muestra en la Tabla 2.1.

Lo primero que hay que calcular es el valor de la cantidad de información I, en base a esto calcularemos la entropía[2] E y la ganancia de información para cada atributo de este ejemplo, estos cálculos vienen descritos a continuación:

$$I(X, C) = -\frac{4}{14} \times \log\left(\frac{4}{14}\right) - \frac{10}{14} \times \log\left(\frac{10}{14}\right) = 0.8631$$

[2]En los calculo se pueden presentar casos en los que tengamos cálculos de la forma $0 \times \log(0)$, por regla de L'Hopital, $a \times log(a)$ vale cero si a es nulo.

Tabla 2.1: Datos de ejemplos para decidir si se juega o no un partido de golf.

Día	*Pronóstico*	*Temperatura*	*Humedad*	*Viento*	*Clase jugar-golf?*
d1	soleado	calor	alta	débil	no
d2	soleado	calor	alta	fuerte	no
d3	nublado	calor	alta	débil	si
d4	lluvia	templado	alta	débil	si
d5	lluvia	frío	normal	débil	si
d6	lluvia	frío	normal	fuerte	no
d7	nublado	frío	normal	fuerte	si
d8	soleado	templado	alta	débil	no
d9	soleado	frío	normal	débil	si
d10	lluvia	templado	normal	débil	si
d11	soleado	templado	normal	fuerte	si
d12	nublado	templado	alta	fuerte	si
d13	nublado	calor	normal	débil	si
d14	lluvia	templado	alta	fuerte	no

$$E(X, \text{Pronóstico}) = (soleado)\frac{5}{14} \times \left(-\frac{2}{5} \times \log\left(\frac{2}{5}\right) - \frac{3}{5} \times \log\left(\frac{3}{5}\right)\right)$$

$$+(nublado)\frac{4}{14} \times (-1 \times \log(1) - 0 \times \log(0))$$

$$+(lluvia)\frac{5}{14} \times \left(-\frac{4}{5} \times \log\left(\frac{4}{5}\right) - \frac{1}{5} \times \log\left(\frac{1}{5}\right)\right)$$
$$= 0.6046$$

$$E(X, Temperatura) = (calor)\frac{4}{14} \times \left(-\frac{1}{2} \times \log\left(\frac{1}{2}\right) - \frac{1}{2} \times \log\left(\frac{1}{2}\right)\right)$$

$$+(templado)\frac{6}{14}\times\left(-\frac{1}{6}\times\log\left(\frac{1}{6}\right)-\frac{5}{6}\times\log\left(\frac{5}{6}\right)\right)$$

$$+(\text{frío})\frac{4}{14}\times\left(-\frac{3}{4}\times\log\left(\frac{3}{4}\right)-\frac{1}{4}\times\log\left(\frac{1}{4}\right)\right)$$

$$=0.7961$$

$$E(X,Humedad)=(alta)\frac{7}{14}\times\left(-\frac{4}{7}\times\log\left(\frac{4}{7}\right)-\frac{3}{7}\times\log\left(\frac{3}{7}\right)\right)$$

$$+(normal)\frac{7}{14}\times\left(-\frac{6}{7}\times\log\left(\frac{6}{7}\right)-\frac{1}{7}\times\log\left(\frac{1}{7}\right)\right)$$

$$=0.7885$$

$$E(X,Viento)=(\text{débil})\frac{8}{14}\times\left(-\frac{6}{8}\times\log\left(\frac{6}{8}\right)-\frac{2}{8}\times\log\left(\frac{2}{8}\right)\right)$$

$$+(fuerte)\frac{6}{14}\times\left(-\frac{4}{6}\times\log\left(\frac{4}{6}\right)-\frac{2}{6}\times\log\left(\frac{2}{6}\right)\right)$$

$$=0.8571$$

- Ganancia de información del atributo *Pronóstico* = $I(X,C)$ - $E(X,\text{Pronóstico})$ = 0.2585
- Ganancia de información del atributo *Temperatura* = 0.067
- Ganancia de información del atributo *Humedad* = 0.0746
- Ganancia de información del atributo *Viento* = 0.0059

Podemos ver que el atributo con valor máximo es el de *Pronóstico*, entonces se procederá a desarrollar el árbol partiendo de este atributo y sus nodos hojas serán los tres valores de este atributo (soleado, nublado y lluvia). De manera recursiva desarrollaríamos el valor soleado con los tres atributos restantes y obtendríamos la siguiente ganancia de información:

- Ganancia de información del atributo *Temperatura* = 0.57
- Ganancia de información del atributo *Humedad* = 0.97
- Ganancia de información del atributo *Viento* = 0.02

Por lo que el máximo en esta ocasión sería el atributo *Humedad* que sería la siguiente partición y expansión del árbol tomando los valores alto y normal. Revisando la Tabla 2.1 podemos ver que ningún valor con la restricción de que es un cielo soleado varía en cada caso, esto es, para las condiciones de humedad alta y cielo soleado todos los ejemplos son de la clase no, y son de la clase si para el caso en el que la humedad es normal y el cielo es soleado, por lo que la parte recursiva en ese instante cumpliría con la condición de paro del algoritmo ID3.

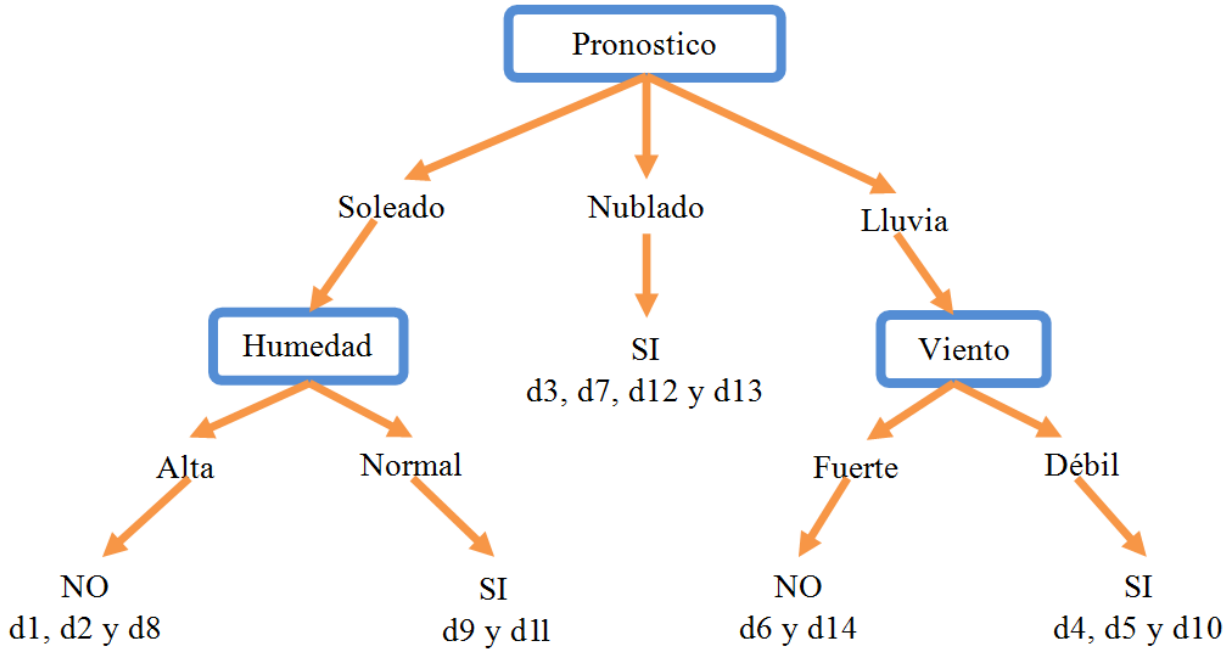
Dado que el algoritmo es recursivo, se regresan a la parte del algoritmo que hizó el llamado con el atributo *Humedad*, en este caso este llamado fue realizado por el valor soleado del atributo *Pronóstico*. El siguiente valor que se revisará es nublado, en el cual podemos ver que todos los ejemplos pertenecen a la clase si, por ello esta parte de ejecución cumple también con la condición de paro del algoritmo ID3.

Posteriormente seguiría el valor lluvia, el cual si varían los ejemplos que tienen este valor, es por ello que se necesita calcular la ganancia de información de los atributos que sobran, obteniendo como resultado al atributo *Viento* como el de mayor cantidad de ganancia de información, por lo que la expansión siguiente sería tomando este atributo y tomando sus dos valores fuerte y débil. Podemos ver que cada ejemplo con el valor de viento fuerte son de la clase no, de la misma forma los ejemplos con valor de viento débil pertenece a la clase si, por lo tanto se cumple la condición de paro del algoritmo ID3 y como ya no hay mas nodos que falten por expandir finalizamos obteniendo el árbol de la Figura 2.3:

2.4. Algoritmo C4.5

El segundo algoritmo para la construcción de árboles de decisión que revisaremos es C4.5[28], este es una mejora de ID3 desarrollado por Quinlan[29] en el año de 1993. Algunas mejoras que presenta C4.5 ante ID3 se listan a continuación:

Figura 2.3: Árbol de decisión resultante para ID3.



- Se manejan atributos contínuos.
- Se mejora la eficiencia computacional.
- Se lleva un control de qué tan profundo va a ser el tamaño del árbol de decisión construido.
- Se evita el sobreajuste (*overfitting*) de datos, esto es que se aprendan a clasificar demasiado bien los datos de prueba, entonces al momento de mostrar ejemplos desconocidos, este no los clasifique de la misma forma que clasifico los ejemplos de prueba.
- Manejo de atributos con diferentes valores.
- Manejo de datos de entrenamiento con valores desconocidos.

El algoritmo C4.5 construye un árbol de decisión a partir de los datos, realizando particiones recursivas de este conjunto y dicha construcción se lleva a cabo mediante la estrategia de primero en profundidad (*depth-first*). Se realizan todas las pruebas posibles para dividir al conjunto de datos

que se tiene y se selecciona aquella que presenta mayor ganancia de información. Para cada atributo discreto (un atributo discreto tiene un número finito o contable de valores) se considera una prueba con n resultados posibles (n es el número de valores posibles que puede tener el atributo), en cambio si se tienen atributos continuos (un atributo continuo tiene un número infinito de valores posibles), se realiza solo una prueba binaria para cada uno de los valores que puede tomar el atributo en los datos. Cada vez que se genere un nodo, el algoritmo debe decidir cuál prueba escoger para ir dividiendo los datos.

Las tres principales pruebas propuestas por el C4.5 son:

1. La prueba llamada *estándar* se utiliza para los atributos discretos, da como resultado una rama por cada valor posible del atributo.

2. Una prueba más compleja, basada en un atributo discreto, en donde los valores posibles son asignados a un número variable de grupos con un resultado posible para cada grupo, en lugar de para cada valor.

3. Si un atributo A tiene valores numéricos continuos, se realiza una con resultados $A \leq Z$ y $A > Z$, por lo cual debemos de determinar el valor límite Z.

Dependiendo del tipo de atributo estas pruebas se van realizando, escogiendo en cada caso a la de mayor ganancia de información.

2.4.1. Funcionamiento de C4.5

El C4.5 se basa en el ID3, por lo tanto, la estructura principal de ambos métodos es la misma, se construye un árbol de decisión mediante el algoritmo de construcción "divide y vencerás", evaluando la información en cada caso utilizando los criterios de Entropía, Ganancia de información, según sea el caso, vistos en la parte de ID3. El árbol de decisión resultante esta formados por:

- *Nodos:* Nombres o identificadores de los atributos.

- *Ramas:* Posibles valores del atributo asociado al nodo.

- *Hojas:* Conjuntos ya clasificados de ejemplos etiquetados con el nombre de una clase.

El pseudocódigo del algoritmo C4.5 para la construcción de un árbol de decisión es el siguiente:

func *C4.5*$(R, C, S) \equiv$ (R: Atributos no clasificados, C: Atributos con clases, S: Ejemplos)

si Sestá vacío,

entonces devolver un único nodo;

otrosi todos los ejemplos son de la misma clase

entonces *C4.5* $\leftarrow$ hoja con clase.

otrosi R está vacío,

entonces Devolver un único nodo con el valor más frecuente del atributo

Clasificador en los registros de S

otrosi

$D\leftarrow$ atributo de mayor Ganancia de Información(D, S)

entre los atributos de R

Sean $d_j, j = 1, 2, ..., m$, los valores del atributo D;

Sean $S_j, j = 1, 2, ..., m$, los subconjuntos de S

correspondientes a los valores de d_j respectivamente;

Devolver un árbol con la raíz nombrada como D

y con los arcos nombrados d_1, d_2,. . .,d_m, que van respectivamente a los árboles

termina

para cada S_i en S

C4.5 $\leftarrow (R - D,$ Clase Mayoritaria, $S_i)$

termina

.

2.4.2. Ejemplo de uso del Algoritmo C4.5

Vamos a tomar el ejemplo de la Tabla 2.1 para poder aplicar C4.5, solo que en este caso quitaremos un valor para ver como C4.5 tiene su funcionamiento con este tipo de situaciones, por lo que los datos para aplicar este algoritmo se muestran en la Tabla 2.2.

Tabla 2.2: Datos de ejemplos para decidir si se juega o no un partido de golf modificada.

Día	*Pronóstico*	*Temperatura*	*Humedad*	*Viento*	*Clase jugar-golf?*
d1	¿?	calor	alta	débil	no
d2	soleado	calor	alta	fuerte	no
d3	nublado	calor	alta	débil	si
d4	lluvia	templado	alta	débil	si
d5	lluvia	frío	normal	débil	si
d6	lluvia	frío	normal	fuerte	no
d7	nublado	frío	normal	fuerte	si
d8	soleado	templado	alta	débil	no
d9	soleado	frío	normal	débil	si
d10	lluvia	templado	normal	débil	si
d11	soleado	templado	normal	fuerte	si
d12	nublado	templado	alta	fuerte	si
d13	nublado	calor	normal	débil	si
d14	lluvia	templado	alta	fuerte	no

La única diferencia esta en el atributo *Pronóstico* ya que se desconoce un valor por lo que la distribución para este atributo seria de la siguiente forma:

	desconocido	*soleado*	*nublado*	*lluvia*
no	1	2	0	2
si	0	2	4	3
totales	1	4	4	5

Procedemos a calcular la cantidad de información de este atributo, sin contar el valor desconocido, por lo que se manejaran solo a 13 de 14 ejemplos posibles, por lo que tenemos lo siguiente:

$$I(S,C) = -\frac{4}{13} \times \log\left(\frac{4}{13}\right) - \frac{9}{13} \times \log\left(\frac{9}{13}\right) = 0.8905$$

Ahora podemos calcular la Entropía del atributo *Pronóstico* con sus diferentes valores, sin contar el valor desconocido:

$$E(S, \textbf{\textit{Pronóstico}}) = (soleado)\frac{4}{13} \times \left(-\frac{2}{4} \times \log\left(\frac{2}{4}\right) - \frac{2}{4} \times \log\left(\frac{2}{4}\right)\right)$$

$$+(nublado)\frac{4}{13} \times (-1 \times \log(1) - 0 \times \log(0))$$

$$+(lluvia)\frac{5}{13} \times \left(-\frac{3}{5} \times \log\left(\frac{3}{5}\right) - \frac{2}{5} \times \log\left(\frac{2}{5}\right)\right)$$
$$= 0.6811$$

Calculamos ahora la ganancia de información del atributo *Pronóstico*, aunque en este caso cambia el calculo ya que contamos solo 13 de 14 ejemplos, por ello tendremos que multiplicar por 13/14 como se muestra a continuación:

Ganancia de información del atributo *Pronóstico* = $\frac{13}{14} \times (I(X, C)$ - $E(X, Pronostico))$ = 0.1458

Ahora definiremos la *división de la Información*, en donde contaremos el atributo que desconocemos de la siguiente manera:

$$I_{division}(\text{Pronóstico}) = -\left(\frac{4}{14} \times \log\left(\frac{4}{14}\right)\right) - \left(\frac{4}{14} \times \log\left(\frac{4}{14}\right)\right)$$
$$-\left(\frac{5}{14} \times \log\left(\frac{5}{14}\right)\right) - \left(\frac{1}{14} \times \log\left(\frac{1}{14}\right)\right) = 1.835$$

Finalmente definiremos una nueva medida la cual lleva por nombre proporción de ganancia de la siguiente manera:

$$\text{proporción\_de\_ganancia}(\textbf{\textit{Pronóstico}}) = \frac{Ganancia(\textbf{\textit{Pronóstico}})}{I_{division}(\textbf{\textit{Pronóstico}})} = 0.098$$

Para los atributos restantes calculamos la ganancia de información y la proporción de ganancia obteniendo los siguientes valores:

Ganancia de información del atributo *Temperatura* = 0.067

$$\text{proporción\_de\_ganancia}(Temperatura) = 0.043$$

Ganancia de información del atributo *Humedad* = 0.0746

$$\text{proporción\_de\_ganancia}(Humedad) = 0.0746$$

Ganancia de información del atributo *Viento* = 0.0059

$$\text{proporción\_de\_ganancia}(Viento) = 0.0060$$

Volvemos a seleccionar el atributo *Pronóstico*, ya sea por su proporción de ganancia o por la ganancia de información como en el caso de ID3, al dividir los 14 ejemplos para construir el árbol, los 13 ejemplos con valor para el atributo *Pronóstico* no presentan problemas y se reparten según sea su valor. Mientras que el caso en que no se conoce el valor, se reparte entre los conjuntos que tienen *soleado*, *nublado* y *lluvia* con los pesos 4/13, 4/13 y 5/13 respectivamente.

Se toma por ejemplo, la división de los datos para el valor *nublado* del atributo *Pronostico*. Los datos que se tienen en cuenta en este caso son:

Día	*Pronóstico*	*Temperatura*	*Humedad*	*Viento*	*Clase jugar-golf?*	*Peso*
d1	¿?	calor	alta	débil	no	4/13
d3	nublado	calor	alta	débil	si	1
d7	nublado	frío	normal	fuerte	si	1
d12	nublado	templado	alta	fuerte	si	1
d13	nublado	calor	normal	débil	si	1

Ahora la distribución de datos para el atributo *Humedad* es:

	desconocido	*alta*	*normal*
no	0	0.3	0
si	0	2	2
totales	0	2.3	2

Los resultados de la ganancia de información y la proporción de ganancia para los atributos restantes son los siguientes:

Ganancia de información del atributo *Temperatura* = 0.068

$$\text{proporción\_de\_ganancia}(Temperatura) = 0.068$$

Ganancia de información del atributo *Humedad* = 0.068

$$\text{proporción\_de\_ganancia}(Humedad) = 0.068$$

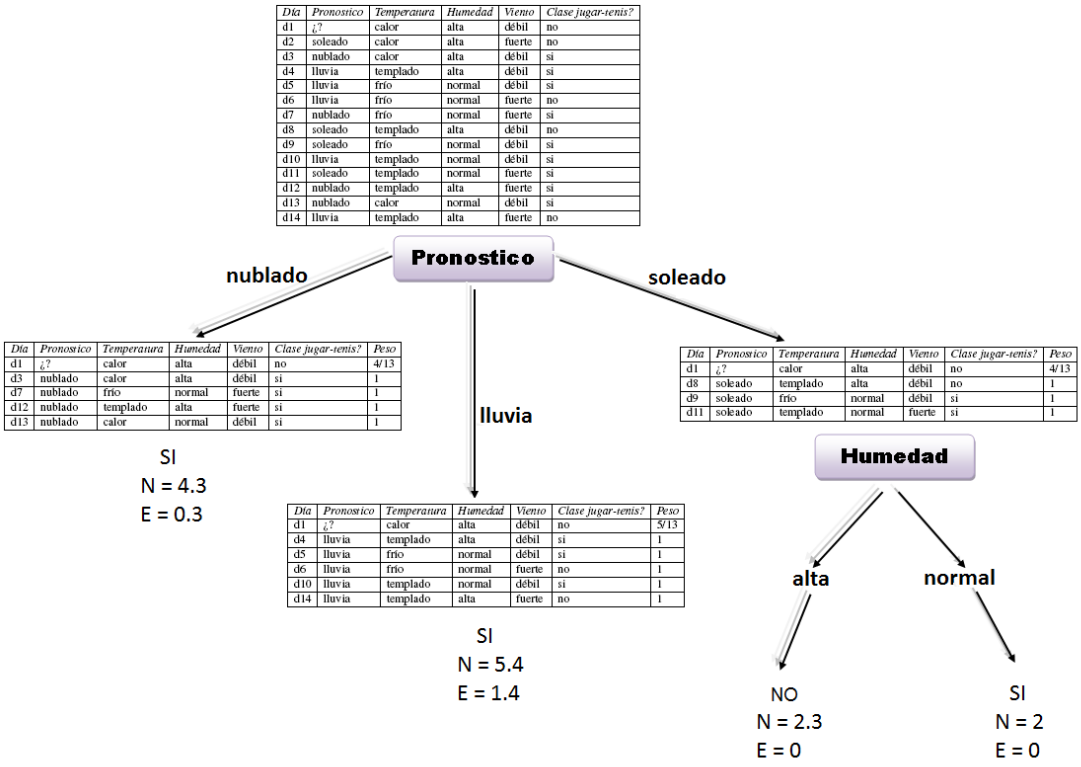
Ganancia de información del atributo *Viento* = 0.068

$$\text{proporción\_de\_ganancia}(Viento) = 0.068$$

Como no se ve ninguna mejora con los diferentes atributos, dejamos de desarrollar el árbol y asignamos la clase *si*, ya que es la clase con mayores ejemplos. La cantidad de casos cubiertos por la hoja, es decir, el *N* asociado es de 4.3. Y la cantidad de casos cubiertos incorrectamente, o el error asociado *E* asociado a esta hoja, es de 0.3.

Finalmente el árbol resultante con C4.5, con la división de datos se muestra en la Figura 2.4.

Figura 2.4: Árbol de decisión resultante para C4.5.



Podemos ver en la Figura 2.4, el árbol generado por el algoritmo C4.5 es muy similar al de ID3 que vimos en la Figura 2.3, a pesar de que en el ejemplo fue omitido un valor para en el ejemplo $d1$, esta es una de las aportaciones que se tienen con C4.5 así como la de reducir el tamaño del árbol, tal fue el caso que para el valor *lluvia* y *nublado* del atributo *Pronóstico*, en donde a diferencia de ID3 no se desarrollo el árbol y se obtuvo una hoja. Las hojas de los árboles de decisión generados por el C4.5 tendrán dos números asociados: N y E. N es la cantidad de casos de entrenamiento cubiertos por la hoja, y E es la cantidad de errores predichos si un conjunto de N nuevos casos fuera clasificados por el árbol. Para obtener dichas hojas, C4.5 analiza los errores predichos en cada uno de los subárboles y ramas del árbol generado para analizar si es conveniente simplificarlo y convertir estos en hojas, de esta manera aplicamos poda en base a los errores predichos. En este

ejemplo, el error total predicho para el árbol estará dado por:

$$Error_{predicho(Arbol)} = 4.3 \times U_{25\,\%}(0.3, 4.3) + 5.4 \times U_{25\,\%}(1.4, 5.4) + 2.3 \times U_{25\,\%}(0, 2.3) + 2 \times U_{25\,\%}(0, 2)$$

Ahora debemos calcular el error total predicho por simplificar el árbol para los valores *lluvia* y *nublado*, asignando la clase *SI*. Esto lo realizamos sumando los valore de N para ambos casos y los dividimos entre la suma de los valores de E, tendríamos entonces:

$$Error_{predicho(Arbol)} = 14 \times U_{25\,\%}(4, 14) = \frac{N_{lluvia} + N_{nublado}}{E_{lluvia} + E_{nublado}} = \frac{9.7}{1.7} = 5.7$$

El error predicho para el árbol simplificado es menor que el error predicho para el árbol generado, es por ello que no desarrollamos las ramas para los valores *lluvia* y *nublado*, comvirtiendolas en hojas y finalizando así este árbol.

2.5. Ventajas y desventajas de los Árboles de Decisión

- Una de las principales ventajas de los árboles de decisión es que su implementación es muy sencilla, por lo que es uno de los métodos más utilizados dentro del aprendizaje inductivo supervisado.
- Una de las desventajas de los árboles de decisión es que siempre se favorece indirectamente a aquellos atributos con muchos valores, los cuales no tienen que ser los más útiles, aunque al aplicar la heuristica Porción de Ganancia de Información (*Information Gain Ratio*), se trata de compensar esta desventaja.
- El algoritmo ID3 utiliza todos los ejemplos del conjunto de entrenamiento en cada paso de su búsqueda guiada, ya que utiliza la noción de la mayor ganancia de información, esto presenta una ventaja, ya que al usar propiedades estadísticas de todos los ejemplos, la búsqueda es menos sensible al ruido que puedan estar presentes en los datos.
- El algoritmo de ID3 no realiza la vuelta atrás (*backtracking*) en su búsqueda, una vez que se selecciona a un atributo, nunca se reconsidera esta selección. Esto puede provocar que

ID3 presente los mismo problemas que los algoritmos que utilizan la estrategia de ascenso a la colina, en concreto que se caiga en mínimos o máximos locales, lo que presenta una desventaja de ID3.

- En los arboles de decisión se presenta el error del sobreajuste (*overfitting*), en donde se aprende a clasificar de una forma bastante bien a los datos de entrenamiento, a tal grado de presentar una generalización de los datos de prueba. Para evitar el problema del sobreajuste se puede implementarse con alguna técnica de poda, la cual es una de las aportaciones de C4.5 con respecto a ID3, esto consiste que en vez de construir un árbol completo, se procede a podar el árbol de manera recursiva para mejorar el rendimiento y así obtener un árbol más corto y fácil de interpretar.

- Otras ventajas que presenta C4.5 respecto a ID3 es, que C4.5 puede manejar atributos continuos, el manejo de datos de entrenamiento con valores faltantes y el manejo de atributos con diferentes valores.

2.6. Análisis de Clusters

2.6.1. Introducción

El término de *clusters*, en español es traducido como conglomerados, aunque en la mayoría de los trabajos que se revisaron al respecto, se utiliza su termino en inglés, es por ello que utilizaremos Análisis de Clusters en vez de Análisis de Conglomerados. El principal objetivo que se tiene en la técnica del análisis de clusters es agrupar un conjunto de datos en un número establecido de clusters o grupos, este agrupamiento se realiza mediante la idea de distancia o similitud entre cada una de las observaciones.

Según sea el criterio de similitud o la distancia escogida, obtendremos diferentes estructuras de clusters. Pongamos de ejemplo el caso en que se tiene una baraja española, en donde se tienen cuatro palos (oros, copas, espadas y bastos), cada palo tiene cartas enumeradas del 1 (representado con As) al 7 y tres cartas con figura (sota, caballo y rey). Si deseamos dividir en un número de clusters este conjunto de datos representado por todas las cartas de la baraja, tendríamos distintos modos de hacerlo: en cuatro clusters (los cuatro palos), en ocho clusters (los cuatro palos y según sean figuras o números), en dos clusters (figuras y números). Es decir, todo depende de lo que consideremos como similar.

Podemos ver que el número de combinación posibles de grupos y de los elementos que integran cada uno de estos grupos, se convierte en un problema muy difícil de resolver por la computadora aún cuando el número de datos no sea muy grande. Es por esto que se desarrollaron métodos que no exploraran el conjunto total de combinaciones, solamente se maneja un subconjunto de estas para encontrar el número de clusters y cuantos componentes tiene cada uno.

Los dos métodos utilizados en el análisis de clusters son los métodos jerárquicos de clusters y los no jerárquicos, los cuales veremos a continuación.

2.6.2. Método jerárquico de clusters

Introducción

La idea básica del método jerárquico de clusters es agrupar un conjunto de observaciones en un número dado de clusters. El enfoque del método jerárquico de clusters tiene dos procedimientos para empezar a trabajar. Uno de ellos es el método jerárquico aglomerativo, en donde se parte con un número de clusters igual al número de objetos o individuos de tal manera que los primeros en hacerlo son los más similares y al final, todos los subgrupos se unen en un único cluster. El segundo método es el llamado método jerárquico dividido, en donde se empieza con un cluster que engloba a todas las observaciones, después se van dividiendo según lo lejanos que estén. En cualquier caso, de ambos métodos se deriva un dendograma[3], que es un gráfico que ilustra cómo se van haciendo las subdivisiones o los agrupamientos, etapa a etapa. Consideramos para este trabajo los métodos aglomerativos con diferentes métodos de unión *(linkage methods)*. Los más importantes son:

- Mínima distancia o vecino más próximo.
- Máxima distancia o vecino más lejano.
- Distancia media.

De este modo, se define una posible distancia entre dos clusters: la correspondiente a la pareja de elementos más cercana, la más lejana y la media de todas las posibles parejas de elementos de ambos clusters.

Medidas de Distancia

Cualquier distancia que utilicemos para medir la similitud entre objetos debe cumplir las siguientes cuatro propiedades[23]:

[3]Un dendograma es un tipo de representación gráfica o diagrama de datos en forma de árbol (del griego *dendro* que significa árbol ó arbustos) que organiza los datos en subcategorías que se van dividiendo en otros hasta llegar al nivel de detalle deseado (asemejándose a las ramas de un árbol que se van dividiendo en otras sucesivamente).

- Dados dos vectores x_i, x_j pertenecientes a $\mathbb{R}^k$, diremos que hemos establecido una distancia entre ellos si definimos una función *d* con las propiedades siguientes:

1. $d : \mathbb{R}^k \times \mathbb{R}^k \to \mathbb{R}^+$, por lo que $d(x_i, x_j) \geq 0$;
2. $d(x_i, x_i) = 0 \ \forall i$, la distancia entre un elemento y sí mismo es cero.
3. $d(x_i, x_j) = d(x_j, x_i)$, la distancia es simétrica
4. $d(x_i, x_j) \leq d(x_i, x_p) + d(x_p, x_j)$, la distancia verifica la propiedad triangular.

Existen muchas distancias que son utilizadas para este tipo de técnicas, definiremos tres distancias que son utilizadas para medir la similitud entre objetos, la primera de ellas es de donde se derivan las otras dos.

La primera distancia que mostraremos es la distancia Minkowski[23], en donde dados dos objetos I_1 y I_2 medidos según dos variables x_1 y x_2, la distancia Minkowski entre ambos es:

$$d_{I_1,I_2} = (|x_{11} - x_{21}|^m + |x_{12} - x_{22}|^m)^{\frac{1}{m}}$$

donde $m \in \mathbb{N}$

Y se puede generalizar con más dimensiones (variables) de la siguiente forma[4]:

$$d_{I_1,I_2} = \sqrt[m]{\sum_{k=1}^{p} (|x_{1k} - x_{2k}|^m)}$$

Si $m = 1$, se tiene la distancia Manhatan y si $m = 2$, la distancia euclideana que son las otras dos distancias que veremos, es por ello que la distancia de Minkowski generaliza a las otras dos.

La distancia euclidiana[23] es la más utilizada cuando los datos son representados en un plano de 2 dimensiones por lo general, aunque se puede dar el caso de que se utiliza esta distancia a representaciones en planos de n dimensiones. Podemos formular esta distancia considerando a dos objetos I_1 y I_2 medidos según dos variables x_1 y x_2 y tomando el coeficiente $m = 2$, la distancia euclidiana entre ambos objetos es:

[4] p representa las dimensiones en las que puede estar representada la distancia.

$$d_{I_1,I_2} = \sqrt{((x_{11} - x_{21})^2 + (x_{12} - x_{22})^2)}$$

Y se puede generalizar con más dimensiones (variables) de la siguiente forma:

$$dI_1, I_2 = \sqrt{\sum_{k=1}^{p}(x_{1k} - x_{2k})^2}$$

Y por último veremos la distancia Manhattan[23], esta distancia hace una similitud con la estructura de las calles de la isla de Manhatan. Esta estructura asemeja una cuadrícula, como los puntos que estarán representados en un plano de una dimensión, se puede utilizar esta medida de distancia representando a cada uno de los puntos sobre una cuadrícula. De la misma forma si se tienen dos objetos I_1 y I_2 medidos según dos variables x_1 y x_2, tomando ahora el valor de $m = 1$ la distancia Manhattan entre ambos es:

$$d_{I_1 I_2} = |x_{11} - x_{12}| + |x_{21} - x_{22}|$$

Y se puede generalizar con más dimensiones (variables) de la siguiente forma:

$$d_{I_1,I_2} = \sum_{k=1}^{p} |x_{1k} - x_{2k}|$$

Funcionamiento del método jerárquico de clusters

Definidas las distancias anteriores, tomamos la distancia euclidiana, que es la mas utilizada para este tipo de clasificadores, para poder enunciar el siguiente algoritmo de construcción de cluster jerárquicos, considerando a N objetos o individuos:

1. Empezar con N clusters (el número inicial de elementos) y una matriz $N \times N$ simétrica de distancias o similitudes. $D = [d_{ik}]_{ik}$.

2. Dentro de la matriz de distancias, buscar aquella entre los clusters U y V (más próximos, más distantes o en media más próximos) que sea la menor entre todas, d_{uv}.

3. Juntar los clusters U y V en uno solo. Actualizar la matriz de distancias:

   - Borrando las filas y columnas de los clusters U y V .
   - Formando la fila y columna de las distancias del nuevo cluster (UV) al resto de clusters.

4. Repetir los pasos (2) y (3) un total de $(N-1)$ veces.

Al final del algoritmo quedaran los objetos divididos en un número de cluster cercano al óptimo y se construirá un dendograma para cada etapa de la ejecución del algoritmo. A continuación se mostrará un ejemplo utilizando el método de unión de mínima distancia.

Ejemplo de uso del método jerárquico de clusters

Sea la matriz de distancias entre 5 objetos dada por:

$$D = [d_{ik}]_{ik} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & & & & \\ 9 & 0 & & & \\ 3 & 7 & 0 & & \\ 6 & 5 & 9 & 0 & \\ 11 & 10 & \mathbf{2} & 8 & 0 \end{bmatrix} \end{array}$$

Cada uno de los objetos comienza siendo un cluster. Como $min\{i,k\} = d_{ik} = d_{53} = 2$ los objetos 3 y 5 se unen para formar el cluster (35). Para construir el siguiente nivel, calculo la distancia entre el cluster (35) y los restantes objetos 1, 2 y 4. Así:

$$d_{(35),1} = min\{d_{31}, d_{51}\} = min\{3, 11\} = 3$$

$$d_{(35),2} = min\{d_{32}, d_{52}\} = min\{7, 10\} = 7$$

$$d_{(35),4} = min\{d_{34}, d_{54}\} = min\{9, 8\} = 8$$

Se reconstruye la matriz de distancias:

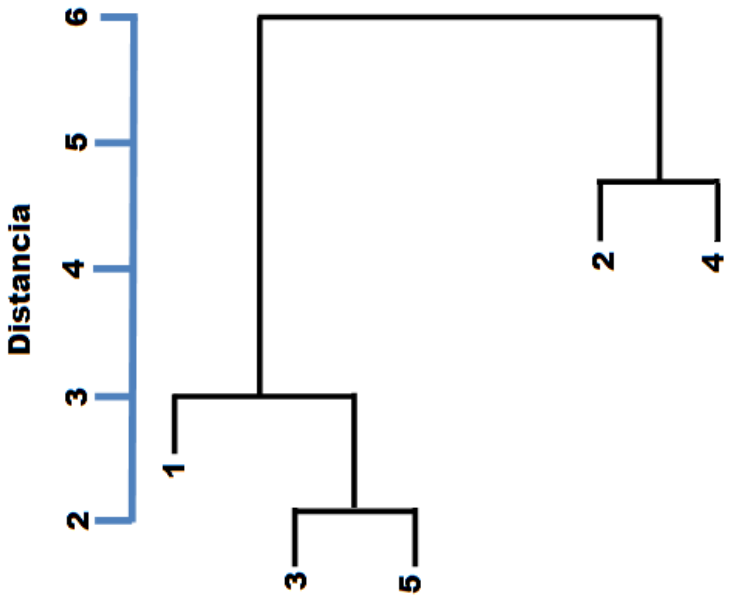
$$D = [d_{ik}]_{ik} = \begin{matrix} \\ (35) \\ 1 \\ 2 \\ 4 \end{matrix} \begin{bmatrix} \begin{matrix} (35) & 1 & 2 & 4 \end{matrix} \\ \begin{matrix} 0 & & & \end{matrix} \\ \begin{matrix} \mathbf{3} & 0 & & \end{matrix} \\ \begin{matrix} 7 & 9 & 0 & \end{matrix} \\ \begin{matrix} 8 & 6 & 5 & 0 \end{matrix} \end{bmatrix}$$

Esto se realizara $N - 1$ veces y obtendremos la siguiente matriz:

$$D = [d_{ik}]_{ik} = \begin{matrix} \\ (351) \\ (24) \end{matrix} \begin{bmatrix} \begin{matrix} (351) & (24) \end{matrix} \\ \begin{matrix} 0 & \end{matrix} \\ \begin{matrix} 6 & 0 \end{matrix} \end{bmatrix}$$

Y finalmente se puede dibujar el dendograma en cada etapa como se muestra en la Figura 2.5:

Figura 2.5: Dendograma de la construcción del cluster por mínima distancia.



Ventajas y desventajas del método jerárquico de clusters

- Los principales problemas que se pueden presentar al usar estas técnicas son los siguientes:
  - Las fuentes de error y variación no entran en consideración con los métodos jerárquicos. Esto implica una gran sensibilidad a observaciones irregulares o extrañas.
  - Si un objeto se ha colocado erróneamente en un grupo al principio del proceso, ya no se puede arreglar en una etapa posterior.
- Un sistema de trabajo conveniente es usar varias distancias o similitudes con los mismos objetos y observar si se mantienen los mismos clusters o grupos. Así, se comprueba la existencia de grupos naturales.

2.6.3. Método no jerárquico de clusters - Método de las k-medias

Introducción

Los métodos no jerárquicos de clusters son utilizados para agrupar conjuntos de objetos u observaciones (en este caso no se pueden agrupar variables) en k clusters previamente establecidos. A diferencia del método de clusters jerárquicos, no se utiliza una matriz de distancias y no se almacenan el número de iteraciones, lo que permite manejar a un número mayor de datos.

Se parte de un número inicial de clusters elegidos de forma aleatoria, en donde se irán depositando cada uno de los objetos de manera iterativa hasta que no se presente una asignación nueva de alguno de los objetos. El método más utilizado es el de las k-medias.

El método de las k-medias permite agrupar al conjunto de observaciones en k clusters, definidos previamente por el usuario, en donde esta asignación se basa en la distancia existente entre cada observación al $centroide$(media) del cluster mas cercano. Con frecuencia la distancia empleada es la distancia euclidiana.

El método de las k-medias es muy utilizada sobre grandes volúmenes de datos. Esta método es utilizada como una técnica exploratoria, en donde se irán clasificando las observaciones en cada uno de los clusters definidos y se irá iterando buscando los centroides de cada cluster para ver si hay

alguna nueva asignación de una observación a un nuevo cluster, este será nuestro criterio de paro, ya que si no existe ninguna nueva asignación en una iteración el clasificador terminará. Cuando se utiliza como una técnica exploratoria no siempre se sabe qué número de k cluster es el mejor, por lo que una técnica para saberlo es realizar pruebas con diferentes k cluster y de esa manera encontrar el mejor. A continuación mostraremos el funcionamiento de este método de las k-medias como una técnica exploratoria.

Funcionamiento del método de las k-medias

Los pasos necesarios para poder desarrollar el método de las k-medias[23] son:

1. Se toman de manera aleatoria k centroides iniciales
2. Para cada una de las observaciones, se calcula la distancia que hay a los centroides y se reasignan a los que estén más próximos. Una vez que se concluye esta reasignación, se vuelven a calcular los centroides de cada uno de los clusters.
3. Se repiten los dos primeros pasos hasta que no se presente una nueva reasignación de alguna observación a uno de los k centroides.

En la practica usualmente se fija el numero de centroides y se procede a realizar el paso 2, esto hace que el ultimo paso cambie, ya que ahora solo se repetirá el paso 2.

Ejemplo de uso del método de las k-medias

Supongamos que se tienen dos variables x_1 y x_2 y además se tienen 4 elementos A,B,C,D con los siguientes valores de cada uno de estos elementos.

	x_1	x_2
A	6	-5
B	2	3
C	6	-2
D	-2	4

Se quiere dividir estos elementos en dos grupos por lo que k lo fijamos a 2. Siguiendo los pasos de construcción del método de k-medias se asignan de manera aleatoria (AB) a un cluster y (CD) al otro y calculamos el centroide de los clusters de la siguiente forma:

- **Cluster** (AB)

$\bar{x_1}$	$\bar{x_2}$
$\frac{6+2}{2} = 4$	$\frac{-5+3}{2} = -1$

- **Cluster** (CD)

$\bar{x_1}$	$\bar{x_2}$
$\frac{6+(-2)}{2} = 2$	$\frac{-2+4}{2} = 1$

Siguiendo con los pasos ya descritos pasamos al (2) y calculamos las distancias euclidianas de cada una de las observaciones a su centroide de acuerdo al cluster en el que estén, si existe algún cambio reasignamos la observación en cuestión al nuevo cluster. Si en efecto existió algún cambio tendremos que volver a calcular los centroides de los clusters que se formen. La distancias euclidianas son las siguientes:

$$d^2(A,(AB)) = (6-4)^2 + (-5-(-1))^2 = 20$$

$$d^2(A,(CD)) = (6-2)^2 + (-5+1)^2 = 32$$

Como A está mas cercano al cluster (AB) que al cluster (CD), no se reasigna.

Se realiza lo mismo para el elemento B:

$$d^2(B,(AB)) = (2-4)^2 + (3-(-1))^2 = 20$$

$$d^2(B,(CD)) = (2-2)^2 + (3-1)^2 = 4$$

Podemos ver que el elemento B se reasigna al cluster (CD), ya que se encuentra a una distancia más cercana, dando lugar al cluster (BCD). A continuación se volverán a calcular los centroides de estos dos nuevos clusters formados:

$Clusters$	$\bar{x_1}$	$\bar{x_2}$
A	6	-5
(BCD)	2	2

Nuevamente, se calculan las distancias para cada observación y de esta manera ver si se producen cambios con respecto a los nuevos centroides:

	A	(BCD)
A	0	65
B	80	1
C	9	32

Se presenta una reasignación del elemento C al cluster (A), por lo que nuevamente se calculan los centroides de los nuevos elementos cluster formados:

$Clusters$	$\bar{x_1}$	$\bar{x_2}$
(AC)	6	-3.5
(BD)	0	3.5

Nuevamente, se calculan las distancias para cada observación y de esta manera ver si se producen cambios con respecto a los nuevos centroides:

	(AC)	(BD)
A	2.25	108.25
B	58.25	4.25
C	2.25	66.25
D	120.25	4.25

Como no se producen cambios, entonces la solución para k = 2 clusters es: (AC) y (BD).

Ventajas y desventajas del método de las k-medias

- Una de las principales ventajas que ofrece este método, es que se pueden manejar volúmenes grandes de datos y los resultados obtenidos suelen ser mejores que los que ofrece el método jerárquico de clusters. Si se desea comprobar la estabilidad de los clusters generados, lo que se realiza es correr de nuevo el algoritmo con otros clusters iniciales.

- La principal desventaja de estos métodos es el valor k de clusters que se escoja, si se impone previamente k clusters puede dar lugar a grupos artificiales o bien a juntar grupos distintos. Una posible solución es considerar distintos números de k de clusters comparando luego sus coeficientes de la *distribución F de Snedecor*[15]. La distribución F de Snedecor se utiliza en la teoría de probabilidad y estadística ya que es una distribución de probabilidad continua. Una variable aleatoria de distribución F de Snedecor se construye como el siguiente cociente:

$$F = \frac{X_1/n_1}{X_2/n_2}$$

En donde:

a) X_1 y X_2 siguen una distribución ji-cuadrada, las cuales presentan n_1 y n_2 grados de libertad respectivamente.

b) Se necesita que tanto X_1 como X_2 son estadísticamente independientes

El objetivo que se persigue al usar la *distribución F de Snedecor* es formar los clusters en donde los centroides estén lo más separados entre sí como sea posible y que las observaciones dentro de cada cluster estén muy próximas al centroide.

2.7. Método de los k-vecinos más cercanos

2.7.1. Introducción

El enfoque de los k vecinos más cercanos es una generalización del método del vecino más cercano. En este tipo de métodos los datos son representados en un plano de n dimensiones en donde previamente se representa un conjunto de ejemplos con sus clases a las que pertenece, para de esta forma poder tener los puntos en el plano para la predicción de un nuevo punto. En la mayoria de los casos no es suficiente tomar solo al vecino más cercano, esto no da una buena clasificación ya que podemos tomar una mala decisión, para ilustrar esto consideremos el siguiente caso:

> Un doctor se dispone a realizar una operación quirúrgica sobre un paciente llamémoslo X. Lo que él puede realizar antes de hacer la operación, es revisar el historial del hospital y encontrar a un paciente que tenga las mismas características del paciente X y que se haya sometido a la misma operación, una vez que se tiene a dicho paciente veremos el resultado de la operación quirúrgica, vemos que fue un fracaso, entonces según este enfoque el doctor decide no operar. Esto es muy drástico ya que en el historial del hospital puede haber un conjunto grande de casos en los que pacientes tal vez no tan similares, pero con características parecidas, se sometieron a la misma operación y resultó que la operación fue todo un éxito. Como vemos estamos dejando fuera a este conjunto de pacientes, que influyen sobre la decisión de realizar o no la operación.

Como vimos en el caso anterior no es muy recomendado el enfoque del vecino más cercano, ya que puede dejar fuera un número considerable de ejemplos los cuales no necesariamente son los más similares, pero que si nos pueden dar una idea mejor de cómo clasificar a cada conjunto datos de prueba. Esta es la justificación que nos lleva a revisar el método de los k vecinos más cercanos, ya que considerando un número k de vecinos se puede obtener una mejor clasificación sobre los datos de prueba.

En el enfoque de los k vecinos más cercanos, lo que se hace es comparar el nuevo dato a ser clasificado con un conjunto de datos de entrenamiento y de este se seleccionan aquellos que

sean más similares, para decidir a qué clase pertenece el nuevo dato. El concepto de similaridad se maneja a través de las distancias que hay entre cada dato, lo que hace que cada uno de estos, contando a los de entrenamiento y prueba, se representen en un plano de n dimensiones (las n dimensiones son de acuerdo al número de variables que describan al objeto) y de esta manera poder ocupar el concepto de distancia. La medida de distancia más utilizada para medir que tan cerca esta un objeto con otro es la distancia euclidiana.

2.7.2. Funcionamiento el método de los *k*-vecinos más cercanos

Teniendo ya definida la distancia métrica euclidiana se puede describir los siguientes pasos para poder desarrollar el método de los *k*-vecinos mas cercanos[23]:

1. Dado un punto x de prueba
2. Encontrar los k vecinos más cercanos sobre los datos de entrenamiento $x_1, x_2, \ldots, x_n$ a x dada la distancia métrica utilizada, en este caso con la que más se trabaja es la distancia euclidiana definida como $d(x, x_i) = \sqrt{((x_{11} - x_{21})^2 + (x_{12} - x_{22})^2)}\ \forall\ i = 1, 2, \ldots, n$
3. En donde cada punto esta denotado por la pareja de coordenadas en n dimensiones de la siguiente forma:
$$(x_{11}, x_{21}), (x_{12}, x_{22}), \ldots, (x_{1k}, x_{2k})$$
4. Dado los k vecinos más cercanos al punto k se hace una votación y se le asigna al punto x la clase que tenga más votos con respecto al número k de vecinos más cercanos que se escoja.

2.7.3. Ejemplo de uso del método de los *k*-vecinos más cercanos

A continuación se mostrará un ejemplo del uso de este método. Se tiene el siguiente conjunto de coordenadas en 2 dimensiones:

Con la clase •:

$$x_1 = (1, 0), x_2 = (2, 1), x_3 = (1, 3), x_4 = (-2, 3),$$

$$x_5 = (-1, -2), x_6 = (4, 2), x_7 = (-1, -1)$$

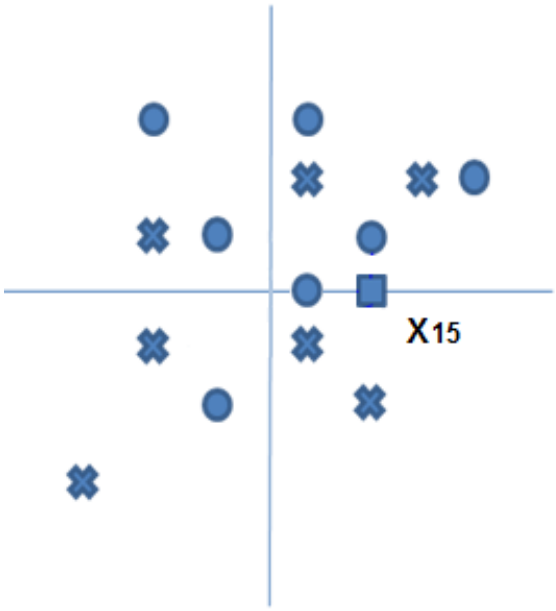
Con la clase $\times$:

$$x_8 = (1, 2), x_9 = (-2, 1), x_{10} = (1, -1), x_{11} = (2, -2),$$

$$x_{12} = (3, 2), x_{13} = (-3, -3), x_{14} = (-2, -1)$$

La representación de los puntos de cada una de las clases se puede ver en la Figura 2.6. Una vez que tenemos estos puntos de entrenamiento el punto de prueba que se quiere clasificar es el $x_{15} = (2, 0)$:

Figura 2.6: Representación de los puntos de entrenamiento.



Se propone el número de vecinos más cercano k = 3, que en la mayoría de los casos siempre es preferible escoger un número impar, para evitar posibles empates, siempre y cunado se trate de 2 clases. Se puede notar que si el valor de k es igual a 1 se tiene el caso del vecino más cercano.

Existen diversos algoritmos que tratan de minimizar el cálculo de distancias hacia todos los posibles datos de entrenamiento y de esta manera reducir el tiempo de cómputo, los más utilizados son el kd-tree[3], algoritmo de Fukunga/Narendra[14], vp-tree[39] y GNAT[6].

Se desea encontrar la clase correspondiente al punto x_{15}, por lo que se localizaran cuáles son sus 3 vecinos más cercanos y después se determinará a que clase pertenece viendo que clase tiene

más representantes. Para el punto x_{15} después de varios cálculos obtenemos que sus tres vecinos más cercanos son x_1, x_2 y x_{10} esto lo obtenemos seleccionando las 3 menores distancias euclidiana al nodo x_{15} de la siguiente forma:

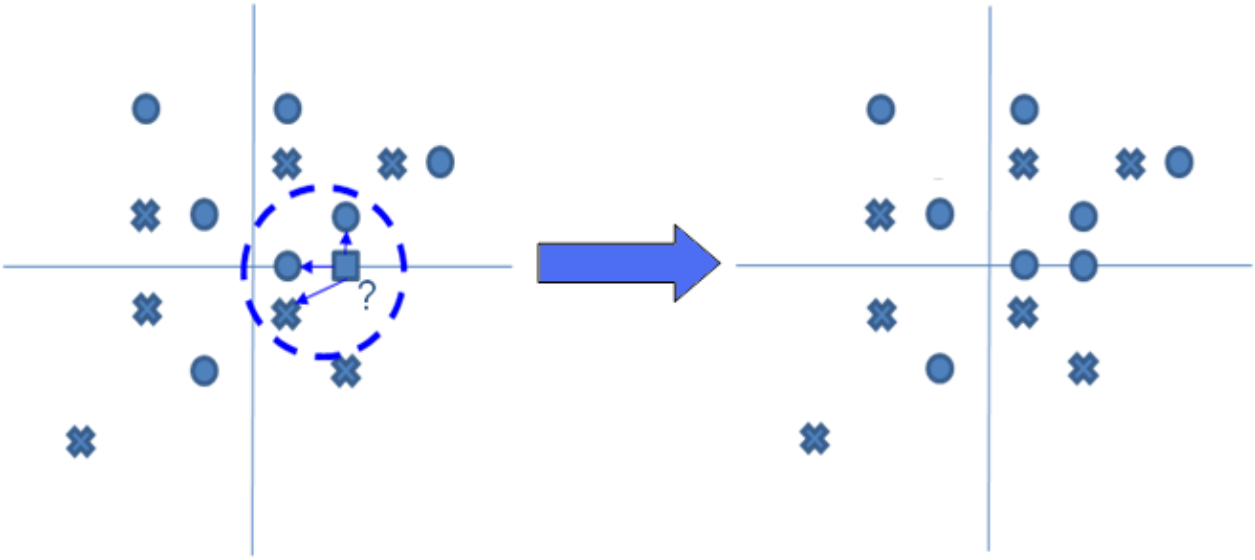
$$d(x_{15}, x_1) = \sqrt{(2-1)^2 + (0-0)^2} = \sqrt{(1+0)} = \sqrt{1} = 1$$

$$d(x_{15}, x_2) = \sqrt{(2-2)^2 + (0-1)^2} = \sqrt{(0+1)} = \sqrt{1} = 1$$

$$d(x_{15}, x_{10}) = \sqrt{(2-1)^2 + (0-(-1))^2} = \sqrt{(1+1)} = \sqrt{2} = 1.41$$

En el plano de dos dimensiones este punto se localiza como se muestra en la Figura 2.7 con sus vecinos más cercanos. Como hay 2 vecinos con la clase • y solo 1 con la clase $\times$, la clase a la que pertenece este nuevo punto x_{15} es •

Figura 2.7: Localización de los vecinos y la clasificación del nuevo punto.



2.7.4. Ventajas y desventajas del método de los *k*-vecinos más cercanos

- La eficacia que pudiera tener el algoritmo depende de la elección de k que hagamos.

- El tamaño de los ejemplos afecta el tiempo total de la clasificación. Por cada nuevo patrón, hay que realizar un calculo de todas las distancias asociadas a este patrón, lo que es muy costoso computacionalmente hablando
- El método se puede aplicar para clases continuas como medias ponderadas
- Existe la posibilidad de ponderar las probabilidades de pertenencia a una clase.

2.8. Clasificadores que utilizan el Teorema de Bayes

2.8.1. Introducción

Como se ha visto en las secciones anteriores, la clasificación de cualquier conjunto de prueba consiste en asignar a cada elemento que conforma este conjunto la clase correspondiente, de acuerdo a varios criterios, como la similitud entre los elementos, los k vecinos más cercanos a este nuevo ejemplo y la construcción de árboles de decisión. Ahora veremos una forma distinta de clasificar a dichos conjuntos de prueba utilizando el Teorema de Bayes, ocupado con mucha frecuencia para desarrollar diversos clasificadores. A continuación se mostraran algunos de ellos empezando desde el clasificador que utiliza la noción de la probabilidad *a priori*, hasta llegar a los clasificadores llamados *Bayes Ingenuo*[22].

2.8.2. Clasificador con probabilidades *a priori*

La primera forma de tratar la clasificación de un conjunto de datos es tomando las probabilidades *a priori*, calculando con la probabilidad muestral, lo cual se hace de la siguiente forma:

Podemos pensar que tenemos un conjunto de entrenamiento de dos clases de peces, una es salmón y la otra es lubina de la forma:

$$X = \{\text{salmón, salmón, lubina, lubina, salmón}, \cdots, \text{ lubina}\}$$

Este conjunto de entrenamiento tiene un tamaño n, entonces para calcular las probabilidades *a priori* de cada una de las dos clases lo que hay que hacer es contar el número en la muestra de

la clase salmón y dividirlas entre el tamaño del conjunto de entrenamiento, de forma similar se hará para las muestras que sean de la clase lubina. Un ejemplo numérico de esto sería tomando un conjunto de entrenamiento de tamaño 100, hay 60 muestras de salmón y 40 muestras de lubina, con el razonamiento de la probabilidad muestral tendríamos las siguientes probabilidades:

La fórmula para calcular la probabilidad *a priori* muestral es:

$$P(Clase_i) = \frac{muestrasClase_i}{n}$$

Sea:

- W_1 = Clase en la que las muestras son $Salmones$
- W_2 = Clase en la que las muestras son $Lubinas$

Las probabilidades de ambas clases serian las siguientes:

- $P(W_1) = 60/100 = 0.6$
- $P(W_2) = 40/100 = 0.4$

Teniendo estas probabilidades *a priori* podemos construir la siguiente regla de clasificación para cualquier dato de prueba que quiera ser clasificado:

Regla de Clasificación con probabilidades *a priori* (2.1):

$$\begin{array}{ll} \textbf{si} & P(W_1) > P(W_2) \\ & \textbf{entonces}\ \ x \text{ es un Salmón} \\ \textbf{otro} & \\ & \textbf{entonces}\ \ x \text{ es una Lubina} \end{array} \tag{2.1}$$

Podemos ver que esta primera aproximación no es muy eficiente ya que tendría una tasa de error de:

$$\text{Tasa de error} = P(W_2) \times 100 = 40\,\%$$

2.8.3. Clasificador con probabilidades condicional

Es por ello que necesitamos tener más información para poder realizar una mejor clasificación. Vamos a suponer que sabemos algo sobre un atributo del pez, por ejemplo; el peso y adicionalmente saber cuál es la probabilidad condicional de que dado un pez con un cierto peso que probabilidad tiene de pertenecer a cualquiera de las dos clases, expresada de la siguiente forma:

$$P(x|W_i)\forall i = 1, 2$$

Con esta información lo que queremos calcular es la probabilidad condicional de que dado un ejemplo nuevo, cual es la probabilidad de que pertenezca a una de las dos clases. Esta probabilidad se define de la siguiente forma:

$$P(W_i|x)\forall i = 1, 2$$

Utilizando el Teorema de Bayes se puede calcular esta probabilidad condicional con la siguiente forma:

$$P(W_1|x) = \frac{P(x|W_j)P(W_j)}{P(x)}$$

Donde $P(x)$ se calcula de la siguiente forma:

$$P(x) = P(x|W1)P(W_1) + P(x|W_2)P(W_2)$$

Toma la Clase W_1 si:

$$\frac{P(x|W_1)P(W_1)}{P(x)} > \frac{P(x|W_2)P(W_2)}{P(x)}$$

Eliminando $P(x)$ tenemos entonces la siguiente regla (2.2):

$$\begin{aligned} &\textbf{si} && P(x|W_1)P(W_1) > P(x|W_2)P(W_2) \\ &&& \textbf{entonces}\ \ x \text{ es un de la clase } W_1 \\ &\textbf{otro} \\ &&& \textbf{entonces}\ \ x \text{ es de la clase } W_2 \end{aligned} \tag{2.2}$$

Además se puede demostrar que esta regla (suponiendo que sabemos las probabilidades *a priori*) reduce mucho la tasa de error con respecto al otro método.

2.8.4. Clasificador con probabilidades condicionales y riesgo condicional

Suponiendo ahora c clases $W_1, W_2, \cdots, W_c$, conocemos las probabilidades *a priori* $P(W_i)$ y las probabilidades condicionales sobre las clases $P(x|W_i)$ para $i = 1, \cdots, c$. Podemos definir el costo de asignar a la observación la clase W_i cuando debería ser de la clase W_j con la siguiente nomenclatura:

$$\lambda(Wi|Wj)$$

Ahora introducimos el concepto de riesgo condicional o pérdida esperada de predecir las clases W_i.

$$R(W_i|x) = \sum_{j=1}^{c} \lambda(W_i|W_j)P(W_j|x) = \lambda(W_i|W_1)P(W_1|x) + \cdots + \lambda(W_i|W_c)P(w_c|x)$$

Por lo que la regla de clasificación bayesiana cambiará, ya que ahora tomaremos la clase que:

$$arg \min_{W_i} R\left[W_i, x\right]$$

Entonces regresando al problema de los peces tendríamos las siguientes expresiones para la perdida esperada:

$$\lambda_{ij} = \lambda(W_i|W_j)$$

$$R(W_1|x) = \lambda_{11}P(W_1|x) + \lambda_{12}P(W_2|x)$$

$$R(W_2|x) = \lambda_{21}P(W_1|x) + \lambda_{22}P(W_2|x)$$

Tomando el caso en el que debemos clasificar un nuevo dato, podemos poner la siguiente regla de clasificación(2.3):

$$A = \begin{cases} R(W_1|x) < R(W_2|x) \wedge \\ \lambda_{11}P(W_1|x) + \lambda_{12}P(W_2|x) < \lambda_{21}P(W_1|x) + \lambda_{22}P(W_2|x) \wedge \\ (\lambda_{21} - \lambda_{11})P(W_1|x) > (\lambda_{12} - \lambda_{22})P(W2|x) \end{cases}$$

Dado un punto x que se desea clasificar, tenemos entonces

$$\begin{array}{ll} \textbf{si} & A \\ & \textbf{entonces} \text{ Toma la clase } W_1 \\ \textbf{otro} & \\ & \textbf{entonces} \text{ tomamos la clase } W_2 \end{array} \tag{2.3}$$

Podemos ver que esta regla (2.3) es mucho más elaborada que la (2.1) y (2.2) vistas anteriormente, pero en ocasiones las probabilidades condicionales $P(W_i|x)$ no son proporcionadas, es por ello que en muchos casos estas son modeladas como distribución gaussiana, esto se verá en la siguiente subsección.

2.8.5. Clasificador con probabilidades condicionales de distribución gaussiana

En esta subsección veremos el caso en el que no tenemos los valores de las probabilidades condicionales. Lo que haremos para calcular estas es modelarlas mediante la distribución gaussiana, que es la forma más usual para clasificadores que hacen uso del Teorema de Bayes. Estas distribuciones están definidas de la siguiente forma:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

La cual sigue una distribución $N(\mu, \sigma^2)$, donde μ es la media y σ es la desviación típica (σ^2 es la varianza)..

Ejemplo: En este solo se muestra una posible manera de representar a las probabilidades *a priori*:

Supongamos que tenemos dos clases, $W_1 = N(0,1)$ y $W_2 = N(1,1)$ y que sus probabilidades *a priori* y condicionales están dadas por:

$$P(W_1) = 0.6\ ,\ P(W_2) = 0.4$$

$$P(x|W_1) = \frac{1}{\sqrt{2\pi}} e^{\frac{-1}{2}x^2}\ ,\ P(x|W_2) = \frac{1}{\sqrt{2\pi}} e^{\frac{-1}{2}(x-1)^2}$$

Podemos observar que para W_1 el valore de $\sigma = 1$ y $\mu = 0$ por ellos la expresión $\left(\frac{x-\mu}{\sigma}\right)^2$ queda como x^2. Para el caso de W_2, $\sigma = 1$ y $\mu = 1$, por lo que la expresión $\left(\frac{x-\mu}{\sigma}\right)^2$ = $(x-1)^2$. La regla que nos daría la clasificación seria la (2.4):

si $R_1 = P(W_1|x) > P(W_2|x)$

entonces Los puntos son asignados a la clase. W_1

otro

Los puntos son asignados a la clase W_2 (2.4)

Como podemos ver falta definir el caso cuando R_1 = R_2, esto de le conoce como la frontera que divide a las dos funciones gaussianas, en este ejemplo el cálculo de esta frontera se lleva de la siguiente forma:

$$P(W_1|x) = P(x|W_1)P(W_1) = \frac{6}{10} \times \frac{1}{2\sqrt{\pi}} e^{\frac{-1}{2}x^2}$$

$$P(W_1|x) = P(x|W_1)P(W_1) = \frac{4}{10} \times \frac{1}{2\sqrt{\pi}} e^{\frac{-1}{2}(x-1)^2}$$

Igualando ambos términos nos queda:

$$3 \times e^{\frac{-1}{2}x^2} = 2 \times e^{\frac{-1}{2}(x-1)^2}$$

Sacando *ln* en ambos lados tenemos:

$$\ln 3 - \frac{1}{2}x^2 = \ln 2 - \frac{1}{2}(x-1)^2$$

$$\ln 3 - \frac{1}{2}x^2 = \ln 2 - \frac{1}{2}(x^2 - 2x + 1)$$

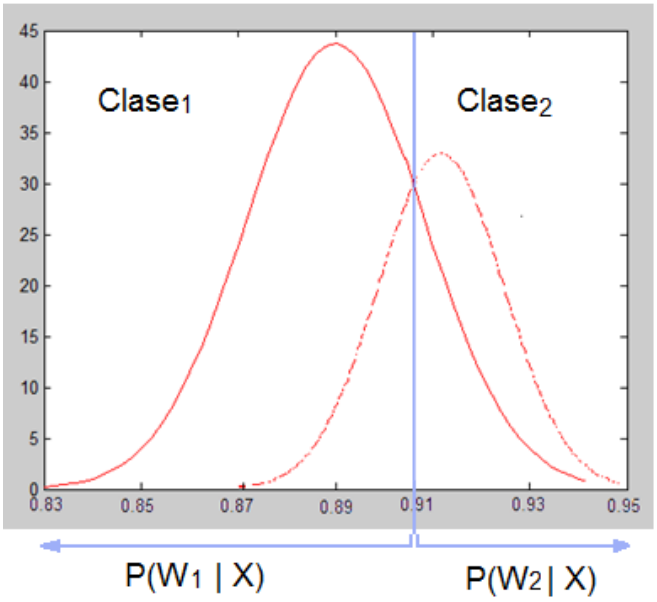
$$\ln 3 - \frac{1}{2}x^2 = \ln 2 - \frac{1}{2}x^2 + x - \frac{1}{2}$$

$$\ln 3 = \ln 2 + x - \frac{1}{2}$$

$$x = \frac{1}{2} + \ln 3 - \ln 2 \approx 0.91$$

Gráficamente se vería como se muestra en la Figura 2.8, en donde podemos observar que aquellos valores a la izquierda del umbral serán asignados a la $Clase_1$ y los valores a la derecha del umbral se les asignara la la $Clase_2$:

Figura 2.8: Gráfica de las probabilidades condicionales de las dos clases y su frontera.



Como podemos ver este método es más completo ya que no necesitamos de tanta información *a priori* para poder hacer una buena clasificación. Este método se puede generalizar para más dimensiones en donde las probabilidades *a priori* $P(W_i|x)$ quedarían expresadas de la siguiente forma:

$$P(x|W_i) = \frac{1}{(2\pi)^{d/2} \times \sqrt{\det(\Sigma)}} \times e^{\left[-\frac{1}{2}\times(x-\mu)^T(\Sigma)^-1(x-\mu)\right]}$$

donde Σ es la matriz de covarianza, explicitamente tenemos:

$$\Sigma_{ij} = \langle x_i x_j \rangle - \langle x_i \rangle \langle x_j \rangle$$

$$\mu_i = \langle x_i \rangle$$

2.8.6. Clasificador Bayes Ingenuo

Otra forma de clasificar haciendo uso del Teorema de Bayes es mediante el Clasificador Bayesiano Ingenuo (*Naive Bayes*), en donde se hace la suposición (que en la mayoría de los ejemplos reales no se presenta) de que todos los atributos son independientes entre dada la clase, de esta manera podemos calcular la probabilidad condicional $P(x|W_i)$ de la siguiente forma[22]:

Sea $X = \{x_1, x_2, x_3, ..., x_d\}$ un vector de prueba al que queremos clasificar cada una de sus componentes y sean dos clases W_1 y W_2, por el Teorema de Bayes y tomando la suposición de que los atributos son independientes entre si dada la clase, podemos expresar la probabilidad condicional $P(X|W_1)$ de la siguiente forma:

$$P(X|W_1) = P(x_1|W_1) \times P(x_2|W_1) \cdots P(x_d|W_1) \times P(W_1)$$

$$P(X|W_1) = P(W_1) \prod_{i=1}^{d} P(x_i|W_1)$$

De forma similar obtenemos la probabilidad condicional $P(X|W_2)$

$$P(X|W_2) = P(x_1|W_2) \times P(x_2|W_2) \cdots P(x_d|W_2) \times P(W_2)$$

$$P(X|W_2) = P(W_2) \prod_{i=1}^{d} P(x_i|W_2)$$

Con estas dos probabilidades condicionales nuestro clasificador bayes ingenuo llevará a cabo su ejecución. El funcionamiento de este clasificador se puede describir en las siguientes dos etapas:

1. La primera es en donde se entrena el clasificador con un conjunto de entrenamiento representado en la mayoría de los casos como una matriz A de $N \times M$, donde N es el número de ejemplos que vamos a tener nuestro conjunto de entrenamiento y M son los atributos que pueden tener cada uno de ellos.

   Este entrenamiento consiste en calcular primero la probabilidad *a priori* de cada una de las clases que estén presentes en el problema, esto se hace calculando cuantos elementos hay de la clase i y dividirlos entre el total que en este caso es N y así para todas las clases que haya en el problema.

   Posteriormente se calculan todas las probabilidades condicionales de cada atributo, si suponemos que existen 2 atributos (a_1 y a_2), los cuales que tienen 2 valores ($v1_{a_1}$ y $v2_{a_1}$) y 3 valores ($v1_{a_2}$, $v2_{a_2}$ y $v2_{a_2}$) respectivamente y que ademas existen 2 clases (W_1 y W_2), el calculo del número de probabilidades condicionales seria un producto de los valores que describen a un atributo por el número de clases. En este caso para el atributo a_1 = 2 × 2 = 4 y para el atributo a_2 = 3 × 2 = 6, en total tendríamos que calcular 10 probabilidades condicionales.

   Después que se calculen todas las probabilidades condicionales, se podrá realizar la clasificación de un nuevo ejemplo, retomando el caso especifico que describimos anteriormente, si el ejemplo nuevo (X) tiene los valores $v2_{a_1}$ y $v3_{a_2}$, para cada atributo debemos de calcula las probabilidades condicionales $P(X|W_1)$ = $P(v2_{a_1}|W_1) \times P(v3_{a_2}|W_1) \times P(W_1)$ y $P(X|W_2)$ = $P(v2_{a_1}|W_2) \times P(v3_{a_2}|W_2) \times P(W_2)$. Finalmente escogemos el argumento máximo de estas dos probabilidades condicionales y asignamos la clase correspondiente al nuevo ejemplo, de esta manera es como se realiza la clasificación con el método Bayes Ingenuo.

2. La segunda fase consiste en ir probando que tan bueno es el clasificador, tomando cada uno de los ejemplos del conjunto de prueba que se utilice. Dado que este método de clasificación maneja probabilidades dentro del intervalo [0,1], al momento de multiplicar las probabilidades condicionales, estas se van haciendo muy pequeñas, por lo que al momento que se implementa sobre una computadora, se tiene un rango limitado de decimales en donde después de alcanzar este umbral, los números dentro de la computadora son interpretados como cero, lo que causaría un problema al llevar la aplicación de este método a la computadora. Para poder evitar que las probabilidades se hagan muy pequeñas incluso se interpreten como cero después de muy pocas operaciones, en vez de multiplicar todas las probabilidades, se saca el logaritmo de cada una de ellas y se cambia la operación de multiplicación por la de suma, en donde se interpretarán mejor los números negativos (dado que se sacarían logaritmos de números entre [0,1]) resultantes.

   Si se tiene las clases a las que pertenece cada ejemplo del conjunto de prueba se puede hacer un estimado del error promedio, comparando cada uno de los ejemplos clasificados con la clase real, cada vez que se encuentra un error se acumulara y después de que se termine la comparación se tendrá el error promedio. En dado caso que no se conozca cual es la clase a la que pertenece se pueden proponer diversos métodos para calcular el error promedio que por el momento no veremos en este trabajo.

Ejemplo de usos del Clasificador Bayes Ingenuo

Los usos para este método de clasificación son muy variados, a continuación se muestran dos ejemplos en donde bayes ingenuo puede funcionar.

- **Clasificación de textos:** Pensemos en que se pretende desarrollar un clasificador de textos de un periódico (El Universal, La Prensa, La Jornada, El Financiero, etc.), se proponen las siguientes clases, que van de acuerdo al contenido de estos periódicos, Deportes, Espectáculos, Política, Cultura, Religión, Sociedad. Lo que se quiere realizar es escoger un fragmento de una clase al azar, presentarlo al clasificador y que este nos diga a que clase pertenece.

La idea de cómo realizar este tipo de clasificadores es tomar a un conjunto de entrenamiento de cada una de las clases, proceder entonces a la etapa de entrenar nuestro clasificador calculando las probabilidades *a priori* y las probabilidades condicionales, las cuales van de acuerdo al número de palabras que se presentan en cada una de las clases, teniendo este dos probabilidades para cada clase. Una vez que se completa la etapa de entrenamiento se procederá a la parte de clasificación de los fragmentos de texto que se ocupen como los datos de entrenamiento, estos serán comparados con los cálculos anteriores y se decidirá a que clases pertenece tomando la que sea mas similar al texto que proporcionemos. La efectividad de este método, a pesar de que se piense que es muy simple, es muy buena ya que presenta buenas clasificaciones.

- **Clasificación de Spam:** Otro uso es en clasificadores de Spam (correo no deseado o correo basura), el cual se encuentran en la mayoría de los gestores de correo para decidir que correo se considera como un correo Spam.

  Una idea básica de cómo poder construir dicho clasificador es tener una base de ejemplos de correos grande, los cuales tengan datos como fecha de envió, nombre del emisor, nombre del correo, hora de envió, etc., además una clase específica, en este caso $Spam$ y $no\ Spam$, que será parte importante para el entrenamiento de nuestro clasificador, de la misma forma que como se ha venido mencionado el funcionamiento de este clasificador se calcularan las probabilidades *a priori* y las condicionales, para que un nuevo ejemplo que sea presentado para ser clasificado use estas probabilidades y de esa manera asignar a que clase pertenece.

  La efectividad de este tipo de aplicaciones no depende solamente de una fase de entrenamiento, si no que necesita estar en un constante entrenamiento para que vaya aprendiendo a diferenciar realmente cuando un correo es o no Spam.

2.8.7. Ventajas y desventajas del Clasificador Bayes Ingenuo

- Una de las principales ventajas de este tipo de clasificadores es que es relativamente sencillo de implementar.

- Suele dar buenos resultados a partir de datos parciales de manera muy rápida y se puede aplicar este tipo clasificadores a Bases de Datos del mundo real obteniendo por lo general buenos índices de clasificación.

- Pero a pesar de ser un buen clasificador, a veces no es muy recomendado usarlo por ello se utilizan métodos más sofisticados como Redes Bayesianas

2.9. Redes Neuronales

2.9.1. Introducción

El Perceptron Multicapa (*Multilayer Perceptron*) es uno de los modelos de Redes Neuronales más utilizados, el cual es una generalización del modelo propuesto por Rosenblatt en 1958[30]. Este modelo establecía un perceptron simple (una sola neurona) para separar por medio de un hiperplano a un conjunto de entrenamiento linealmente separable. En 1986 Rumelhart y McClelland[25] demostraron que algunos problemas imposibles para los perceptrones simples pueden ser resueltos por redes multinivel con funciones de activación no lineales, usando procesos simples de entrenamiento.

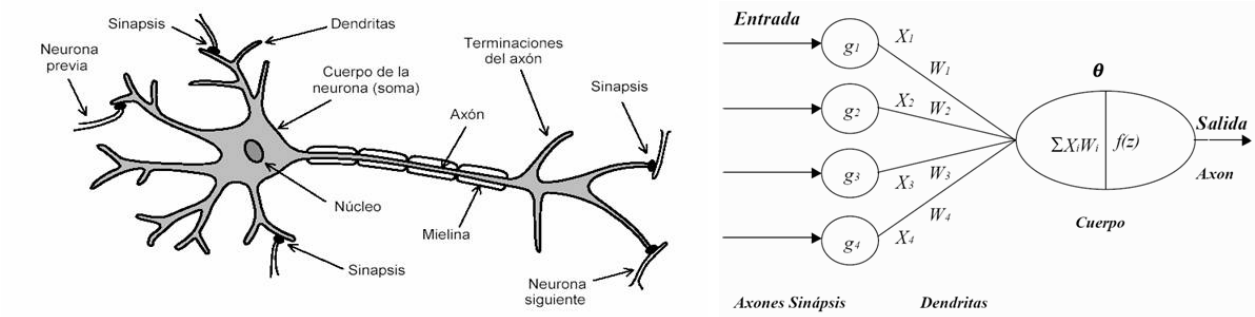
En el caso de los seres humanos dentro del cerebro existe un número $\approx 10^{10}$ de neuronas y con $\approx 10^4$ interconexiones entre cada una de ellas. A nivel del funcionamiento de cada neurona, este es muy lento comparado con un procesador (las neuronas realizan 10 operaciones por segundo mientras que los procesadores realizan millones de operaciones por segundo).

Las partes que conforman a una neurona son tres: las entradas llamadas *dendritas*, el cuerpo mismo de la neurona y las salidas llamadas axones. En la parte final de cada axón se encuentra un elemento que permite la comunicación con mas *dendritas* de otra neurona, llamado *sinapsis*. Las neuronas aceptan miles de señales de entradas con una fuerza determinada, dependiendo de estas la neurona emite una señal de respuesta, por lo que las *sinapsis* pueden entonces trasmitir una señal débil o fuerte dependiendo de la fuerza que haya salido del procesamiento de la neurona. Desde un enfoque matemático el funcionamiento de una neurona puede representarse por una lista de sus

señales de entrada que son multiplicadas por sus pesos correspondientes y después son sumadas cada una de las entradas. El resultado es el nivel de activación de la neurona, que es la entrada hacia las demás neuronas que se encuentren conectadas a ella.

El modelo del perceptron simple es uno de muchas representaciones de una red neuronal artificial, el cual trataba de imitar el comportamiento de una neurona, en donde se tiene un conjunto de entradas (*axones simples*) las cuales son conectadas al cuerpo del perceptron (*cuerpo de la neurona*) a través de un conjunto de vértices (*dendritas*). Dentro del cuerpo del perceptron se encuentra una función que suma cada una de las entradas y la función de activación de la neurona. En la Figura 2.9 se muestra el esquema de una neuronal real y el esquema de un Perceptron Simple.

Figura 2.9: a) Modelo real de una neurona, b) Modelo del perceptron simple.

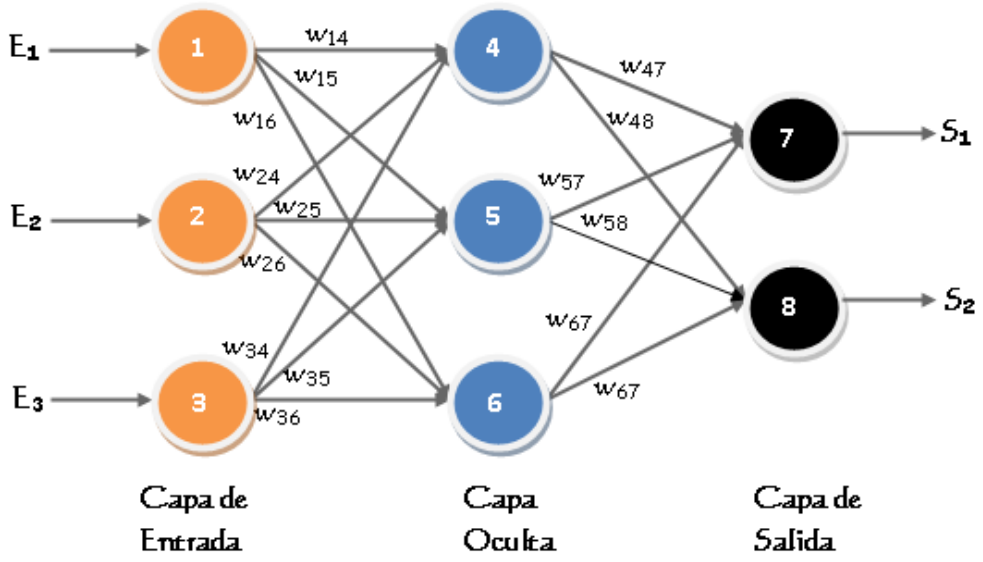


Tanto el perceptron simple como el perceptron multicapa (PMC) son representaciones de una red neuronal artificial. Cuando se unen varios perceptrones en una capa y los estímulos recibidos en las entradas de cada uno son sumados, se considera que se tiene una red neuronal. El enfoque del PMC es precisamente combinar varios perceptrones simples en un esquema donde las salidas de cada uno, son transmitidas varias capas hasta llegar finalmente a las salidas de los PMC.

La estructura de PMC está dividido por capas las cuales son: la capa de entrada, la(s) capa(s) oculta(s) y la(s) capa(s) de salida(s). En este tipo de red neuronal artificial se ingresa un número de entradas que van conforme al número de neuronas que hay en la capa de entrada, después estas se conectan con las neuronas de la capa oculta por medio de vértices, en donde cada salida de las neuronas presentes en la capa de entrada, se asocia a cada una de las neuronas en la capa oculta. Una vez que están conectadas todas las salidas de la capa de entrada a las entradas de las neuronas

de la capa oculta se repite el mismo procedimiento en caso de que se presenten más capas ocultas, si no es el caso las salidas de cada neurona en la capa oculta es conectada a la capa de salida, de la misma forma que se describió, teniendo finalmente las salidas finales del PMC. En la Figura 2.10 se muestra el esquema básico del PMC:

Figura 2.10: Modelo del Perceptron Multicapa - Retropropagación.



Dado que en las décadas de los 60'ss y 70's no hubo métodos de entrenamiento apropiados para los PMC, se vio detenido el interés por estas tipologías. Fue hasta mediados de los 80's cuando Rumelhart[25] propuso un método de entrenamiento para los PMC llamado retropropagación (*backpropagation*), el cual será descrito a continuación.

2.9.2. Funcionamiento del Perceptron Multicapa - Retropropagación

El termino retropropagación se basa en el método del gradiente descendiente para encontrar el error en una red hacia adelante (*feed-foward*, de aprendizaje supervisado, en donde se necesita un conjunto de entrenamiento y el valor o meta esperada), que es una aplicación directa de la Regla de la Cadena utilizada en Calculo Diferencial [20].

El funcionamiento de este tipo de redes neuronales artificiales se puede dividir en las siguientes

dos etapas:

1. Los datos de entrenamiento se pasan hacia delante (*forward pass*), las salidas son calculadas calculando el error en cada caso.

2. Se realiza entonces el paso hacia atrás (*backward pass*), en donde el error calculado en la capa de salida, se utiliza para cambiar el peso da cada capa ocultas de la red neuronal, hasta llegar a la capa de salida, calculando recursivamente los gradientes locales para cada neurona. Al final de estas dos etapas se tiene un PMC entrenado.

A continuación se muestra el algoritmo básico de construcción de un PMC:

1. Determinar la Arquitectura.

   - Cuantas unidades de entrada y salida.
   - Cuantas capas ocultas y unidades en cada capa oculta.

2. Inicializar todos los pesos y sesgos a valores aleatorios pequeños por ejemplo $\in$ [-1,1] y el valor de η[5] .

3. Repetir hasta que el criterio de terminación sea satisfecho

   - Presentar un ejemplo de entrenamiento y pasarlo por la red (*forward pass*)
   - Calcular la salida actual y el error en cada salida
   - Adaptar los pesos empezando por la capa de salida y trabajar hacia atras (*backward pass*).

En donde se tiene:

$$W_{pq}(t+1) = W_{pq}(t) + \Delta W_{pq} \rightarrow W_{pq}(t) \text{ Peso del nodo } p \text{ al nodo } q \text{ en el tiempo } t$$

[5]Constante utilizada para el valor de ajuste, donde se tienen dos criterios para escogerla, valores muy pequeños: hacen una convergencia lenta, valores muy grandes: riesgo de *overshooting* (saltarnos al mínimo local)

$$\Delta W_{pq} = \eta \cdot \delta_q \cdot O_p \rightarrow \text{Cambio de pesos}$$

$$\delta_i = (d_i - O_i) \cdot O_i \cdot (1 - O_i) \rightarrow \text{Para cada unidad de salida de la neurona } i$$

$$\delta_j = O_j \cdot (1 - O_j) \cdot \sum_i W_{ij} \cdot \delta_i$$

En donde para cada unidad oculta j, y la suma sobre todos los nodos i en la capa anterior j, O_i son las salidas obtenidas y δ_i son las salidas deseadas.

Es necesario proponer una función f sigmoide (la función de activación en el caso del perceptron simple), que sea diferenciable. La función sigmoide es una de las funciones de transferencia más utilizadas. Produce salidas continuas y proporcionales al nivel de activación de la neurona dentro del rango [0,1]; sus niveles de saturación son 0 y 1, por lo que su salida máxima será 1 y la mínima 0. Cuando el nivel de activación supere al umbral de saturación máximo la salida seguirá siendo 1 y cuando el nivel de activación sea inferior al umbral de saturación mínimo la salida seguirá siendo 0. Es común tomar la función sigmoide exponencial denotada por:

$$f(x) = \frac{1}{1 + e^{-x}}$$

Cuya derivada es:

$$f'(x) = \frac{1}{(1 + e^{-x})^{2e^{-x}}}$$

Aunque también se usan otras como la tangente hiperbólica:

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

o la identidad:

$$f(x) = x$$

Si escogemos la función sigmoide exponencial tendríamos el siguiente criterio:

si $f(x) = \frac{1}{1+e^{-x}}$
entonces $f'(net) = f(net) \cdot (1 - f(net)) = O_q \cdot (1 - O_q)$

Usualmente se utilizan tres criterios de paro para este tipo de métodos, los cuales son:

1. Número de épocas (se le llama época al proceso de entrenar 1 vez el perceptron multicapa sobre todos los ejemplos.)
2. Error Mínimo Cuadrado, esto se realiza al llevar acabo el entrenamiento, en donde se tiene un registro de los errores que se van presentando en cada una de las épocas y se decide parar cuando se encuentre un error mínimo.

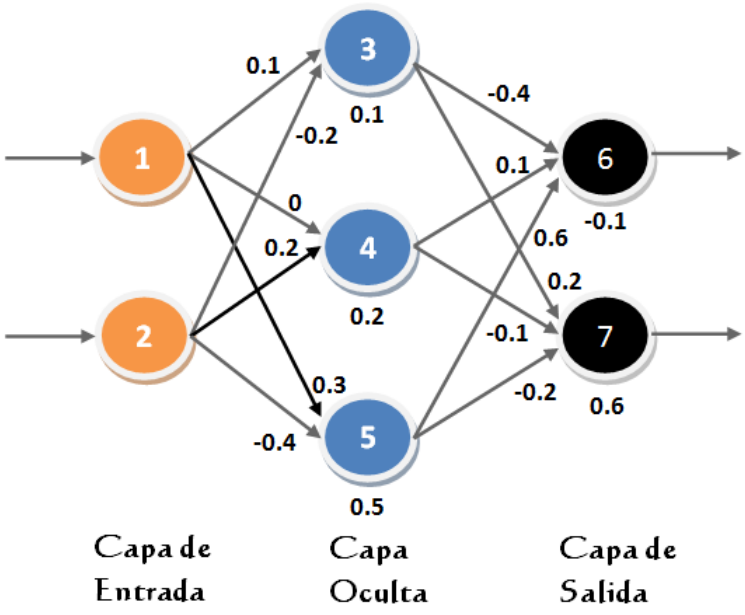
En cuestión de las funciones de activación tenemos las siguientes opciones:

1. Diferenciables (el método propuesto de retropropagación)
2. No lineales
3. Monotónica (una función que siempre crece o decrece)
4. Lineal para valores pequeños.
5. Asimétricas

2.9.3. Ejemplo de uso del Perceptron Multicapa - Retropropagación

La siguiente PMC tiene dos unidades en la capa de entrada, tres unidades en la capa oculta, y dos unidades en la capa de salida, se dice que esta es la arquitectura o la topología de la red. Los valores junto a las unidades 3, 4, 5, 6, 7 (es decir, 0.1, 0.2, 0.5, -0.1, 0.6) son sesgos que también están actualizados usando retropropagación. Supóngase que la función de activación es el sigmoide exponencial: $f(x) = \frac{1}{1+e^{-x}}$. La topología de este PMC se muestra en la Figura 2.11.

Figura 2.11: Topología del PMC para el ejemplo.



Si se presenta una entrada (0.6, 0.1), con salida deseada de (1 ,0), se procederá a calcular las salidas en cada una de las unidades 3, 4, 5, 6 y 7, después se calcularan los errores y las actualizaciones de todos los pesos y sesgos, para tener un PMC entrenado con esta entrada.

Primero lo que debemos de hacer es cambiar esta estructura del perceptron y agregarle sesgos en la capa de entrada y la capa oculta, los cuales tienen el efecto de disminuir la entrada de la función de activación de la neurona, por lo que la arquitectura de nuestro perceptron multicapa quedaría como se muestra en la Figura 2.12.

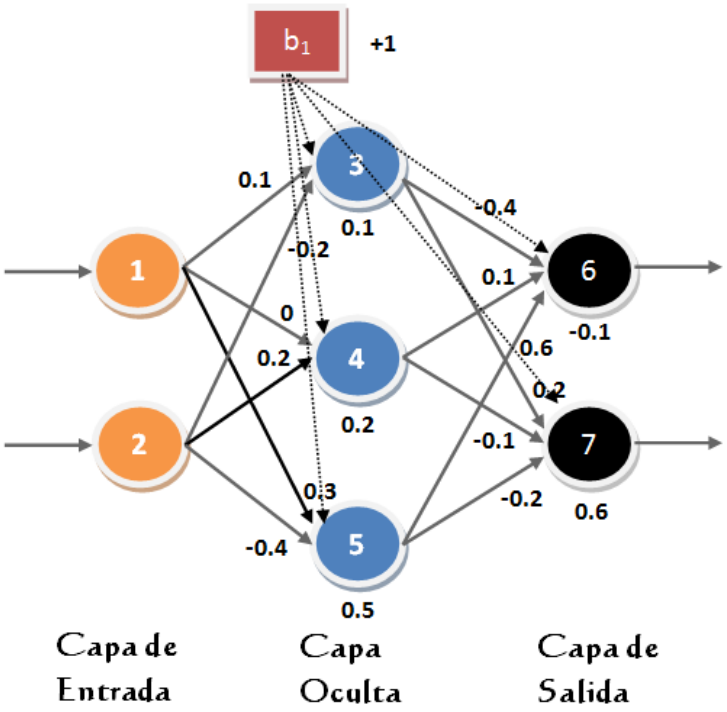
Lo que sigue por realizar es presentar el ejemplo con la entrada z^1 = (0.6, 0.1) y propagarlo por la red para poder encontrar su salida real. La entrada z^1 cambia a z^1 = (1, 0.6, 0.1) por el sesgo b_1 por lo que tendríamos:

Para el nodo 3:

$$z^1 = (1, 0.6, 0.1)$$

$$w = (0.1, 0.1, -0.2)$$

Figura 2.12: Topología del PMC para el ejemplo agregándole los sesgos.



En este caso vemos que tomamos el vector w considerando el sesgo que hay presente en el nodo 3 que es igual a 0.1 y las entradas que recibe del nodo 1 y el nodo 2 con los valores 0.1 y -0.2 respectivamente.

$$n = \langle z^1, w \rangle = (1 * 0.1 + 0.6 * 0.1 + 0.1 * -0.2)$$

$$= (0.1 + 0.06 - 0.02) = 0.14$$

$$O_3 = f(0.14) = 0.53$$

Para el nodo 4:

$$z^1 = (1, 0.6, 01)$$

$$w = (0.2, 0, 0.2)$$

$$n = \langle z^1, w \rangle = (1 * 0.2 + 0.6 * 0 + 0.1 * 0.2)$$

$$= (0.2 + 0 + 0.02) = 0.22$$

$$O_4 = f(0.22) = 0.55$$

Para el nodo 5:

$$z^1 = (1, 0.6, 01)$$

$$w = (0.5, 0.3, -0.4)$$

$$n = \langle z^1, w \rangle = (1 * 0.5 + 0.6 * 0.3 + 0.1 * -0.4)$$

$$= (0.5 + 0.18 - 0.04) = 0.64$$

$$O_5 = f(0.64) = 0.65$$

Ahora como la entrada hacia la capa oculta ha cambiado se necesita actualizar z^1 por z^2 = (1, 0.53, 0.55, 0.65), por lo que tendríamos para los siguientes nodos los siguientes resultados:

Para el nodo 6:

$$z^2 = (1, 0.53, 0.55, 0.65)$$

$$w = (-0.1, 0.4, 0.1, 0.6)$$

$$n = \langle z^2, w \rangle = (1 * -0.1 + 0.53 * 0.4 + 0.55 * 0.1, 0.65 * 0.6)$$

$$= (-0.1 + 0.212 + 0.055 + 0.39) = 0.133$$

$$O_6 = f(0.133) = 0.53$$

Para el nodo 7:

$$z2 = (1, 0.53, 0.55, 0.65)$$

$$w = (0.6, 0.2, -0.1, -0.2)$$

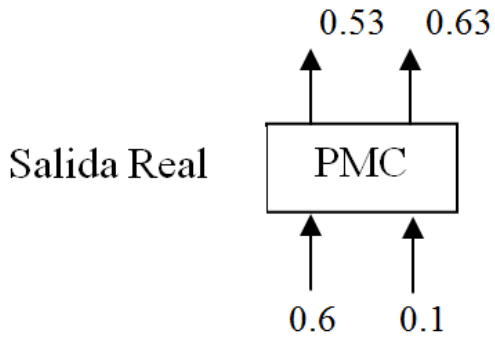
$$n = \langle z2, w \rangle = (1 * 0.6 + 0.53 * 0.2 + 0.55 * -0.1, 0.65 * -0.2)$$

$$= (0.6 + 0.106 - 0.055 - 0.13) = 0.521$$

$$O_7 = f(0.521) = 0.63$$

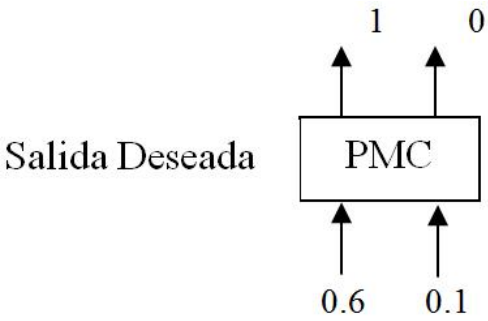
Por lo que nuestra salida real con la entrada (0.6, 0.1) se ve en la Figura 2.13:

Figura 2.13: Salida real del PMC con las entradas propuestas.



Ahora realizamos el paso hacia atrás calculando los errores, los valores de δ, empezando por la capa de salida y después ir a cada una de las capas de la red. La Figura 2.14 muestra la salida deseada:

Figura 2.14: Salida deseada del PMC.



Para las q's unidades en la capa de salida tenemos:

$$\delta_q = (\delta_q - O_q)O_q(1 - O_q)$$

Para el nodo 6 tenemos:

$$\delta_6 = (\delta_6 - O_6)O_6(1 - O_6)$$

$$= (1 - 0.53)0.53(1 - 0.53) = 0.12$$

Para el nodo 7 tenemos:

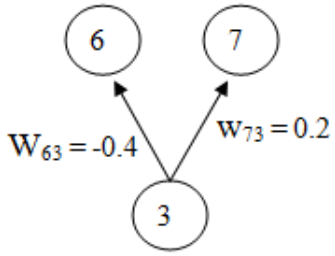
$$\delta_7 = (\delta_7 - O_7)O_7(1 - O_7)$$

$$= (0 - 0.53)0.63(1 - 0.63) = -0.15$$

Para cada una de las q's unidades en la capa oculta, retropropagamos los valores de las δ:

$$\delta_q = O_q(1 - O_q)\sum_i W_{iq}\delta_i$$

Para el nodo 3 tenemos:



$$\delta_3 = O_3(1 - O - 3)[w_{63}\delta_6 + w_{73}\delta_7]$$

$$= 0.53(1 - 0.53)[(-0.4)0.12 + 0.2(-0.15)] = -0.019$$

$$\delta_4 = 0.01$$

$$\delta_5 = 0.02$$

Ahora lo que debemos de hacer es actualizar los pesos de todas las capas con la siguiente fórmula:

$$W_{qp}^{nuevo} = W_{qp}^{viejo} + \Delta W_{qp}$$

Nuestra η = 0.1, por lo que tendríamos para cada uno de los pesos:

Para el peso W_{63} tenemos:

$$W_{63}^{nuevo} = W_{63}^{viejo} + \Delta W_{63} = W_{63}^{viejo} + \eta\delta_6 O_3$$

$$= -0.4 + (0.1)(0.12) * (0.53) = -0.39$$

Para el peso W_{73} tenemos:

$$W_{73}^{nuevo} = 0.19$$

Para el peso W_{64} tenemos:

$$W_{64}^{nuevo} = 0.11$$

Para el peso W_{74} tenemos:

$$W_{74}^{nuevo} = 0.11$$

Para el peso W_{65} tenemos:

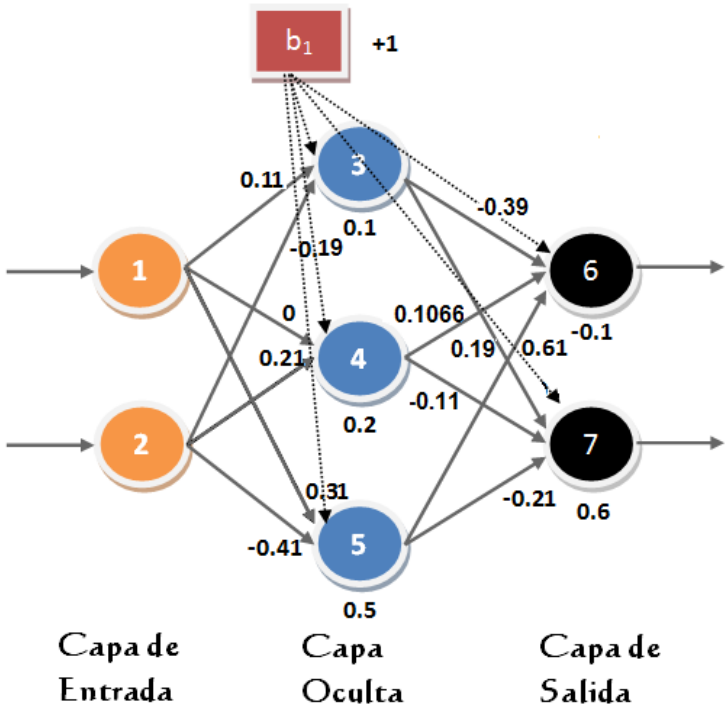
$$W_{65}^{nuevo} = 0.61$$

Para el peso W_{75} tenemos:

$$W_{75}^{nuevo} = 0.21$$

Hasta este punto los pesos para la capa oculta han sido actualizados, esto se realiza para la siguiente capa hasta finalizar obteniendo un PMC entrenado con una entrada (0.6, 0.1) tal y como se muestra en la Figura 2.15.

Figura 2.15: PMC entrenado con la entrada (0.6, 0.1).



2.9.4. Ventajas y desventajas del Perceptron Multicapa - Retropropagación

- Las redes perceptron multicapa (también denominadas redes de retropropagación, por el método de entrenamiento), son una buena técnica dentro del aprendizaje supervisado (utilizan para su entrenamiento patrones formados por un vector de entrada y un vector de salida).
- Suelen ocuparse estos métodos con diferentes condiciones de paro y funciones de activación para probar la eficiencia en cada caso.
- Una desventaja que resentan los PMC es el problema de la generalización, en el cual el PMC aprende correctamente los datos de entrenamiento, pero no es capaz de responda de forma adecuada ante datos nuevos. Para solucionar esto, es necesario que durante el aprendizaje el PMC extraiga las características de las muestras que le permitan responder correctamente a patrones diferentes.

- Las superficies que definen el error calculado en cada salida es muy compleja y ruidosa. Debido a que se utiliza el método del gradiente descendiente se corre el riesgo de quedar atrapado sobre mínimos locales. Una forma de evitar estos mínimos locales es aumentar el número de neuronas en las capas ocultas, ya que los PMC son dependientes de el número que capas ocultas.

- Se puede presentar el problema de parálisis, este fenómeno también se conoce como saturación, y se produce cuando la entrada total a una neurona de la red toma valores muy altos, tanto positivos como negativos, dado que las funciones de activación poseen dos asintotas horizontales, si la entrada alcanza un valor alto, esta se satura y alcanza un valor de activación maximo o mínimo. Es por ello que se recomienda partir de valores iniciales aleatorios próximos a cero.

2.10. Reglas de Decisión

2.10.1. Introducción

Una de las formas mas utilizadas, junto con árboles de decisión, para la clasificación de datos es la que se lleva a cabo mediante Reglas de decisión. Existen diferentes formas de construir un modelo de clasificación basado en reglas de decisión:

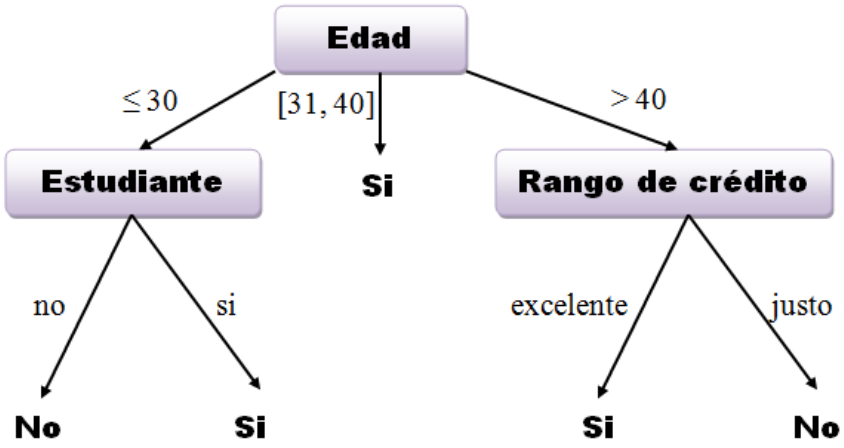
- **Árboles de Decisión:** En este tipo de estructuras se crea un conjunto de reglas que son más fáciles de interpretar que un árbol complejo, se crea una regla por cada hoja del árbol.

  Podemos tener un ejemplo sencillo de como crear estas reglas a partir del árbol de decisión para otorgar un crédito, este se puede ver en la Figura 2.16.

  Estos modelos que se construyen a partir de un árbol de decisión puedes ser simplificados (generalizados), pero pueden perder las dos características esenciales:

  - Se pierde la característica de ser mutuamente excluyentes ya que varias reglas pueden ser válidas para un mismo ejemplo, por lo que se deben de establecer un orden entre las

Figura 2.16: Reglas de decisión a partir de un árbol.



SI (edad ≤ 30) & (estudiante == no) ENTONCES compras_por_computadora = No
SI (edad ≤ 30) & (estudiante == si) ENTONCES compras_por_computadora = Si
SI (30 < edad ≤ 40) ENTONCES compras_por_computadora = Si
SI (edad > 40) & (rango_de_credito == excelente) ENTONCES compras_por_computadora = Si
SI (edad > 40) & (rango_de_credito == justo) ENTONCES compras_por_computadora = No

reglas (lista de decisión) o proponer un criterio de votación.

- Se pierde la característica de ser exhaustivos, ya que se puede presentar el caso en el que ninguna regla generada sea aplicable a un ejemplo concreto, por lo que se debe de incluir una clase por defecto.

- **Algoritmos específicos de Inducción de reglas:** El lenguaje de representación de las Reglas de Decisión es esencialmente proposicional. Es este tipo de modelo, cada regla se aprende de una en una, por lo que que cada vez que se selecciona una regla, se eliminan del conjunto de entrenamiento todos los ejemplos que son cubiertos por la regla seleccionada. El proceso se repite iterativamente hasta que se alguna condición de paro se cumple. Para poder aprender una regla se empieza con reglas lo más general posible, después a estas se les van añadiendo antecedentes para maximizar la precisión de clasificación de esta regla.

- **Modelos basados en reglas de asociación:** El objetivo de las reglas de asociación es encontrar asociaciones o correlaciones entre los elementos u objetos presentes en las bases de

datos. Se buscan las mejores reglas de asociación, para poder superar algunas limitaciones de los modelos basados en árboles de decisión, los cuales consideran sólo los atributos de uno en uno de manera parcial.

2.10.2. Tablas de Decisión

Muchos procesos de toma de decisiones pueden ser tratados por medio de Tablas de Decisión[24], en las que se representan los elementos característicos de estos problemas:

- Las diferentes *combinaciones de valores* que puede presentar la naturaleza denotados por $v_1, v_2, \ldots, v_n$. En donde cada $v_i \; \forall i = 1, \ldots, n$, tiene un valor especifico.
- Existen ademas *causas* o *condiciones* que propician cada uno de los estados de la naturaleza los cuales los denotamos por $c_1, c_2, \ldots, c_n$.
- Y por ultimo tenemos los *efectos* o *acciones* que se seleccionaran: $a_1, a_2, \ldots, a_m$.

En la Tabla 2.3 se muestra el formato general de una Tabla de Decisión, esta tabla contiene m causas que se presentan en el problema denotadas por c_1, c_2, ..., c_m. Por cada causa se tiene n combinaciones posibles, y en cada combinación se pueden tomar distintos valores denotados por $V_i \; \forall \; i = 1, \ldots, n$ y finalmente se tienen dos acciones, a_1 y a_2.

En este tipo de representación matricial, se pueden representar todas las situaciones posibles asociadas a una problemática presente en una BD, junto con las acciones que se deben de tomar para cada situación.

Tabla 2.3: Formato general de una Tabla de Decisión.

		Combinaciones			
Causas	*Valores*	v_1	v_2	...	v_n
c_1	V_1	x_{11}	x_{12}	...	x_{1n}
c_2	V_2	x_{21}	x_{22}	...	x_{2n}
c_i	...	...	...	...	...
c_m	V_n	x_{m1}	x_{m2}	...	x_{mn}
Acciones					
a_1		$\times$		...	$\times$
a_2			$\times$	...	$\times$

2.10.3. Funcionamiento de Tablas de Decisión

Los pasos para llevar a cabo la construcción de las Tablas de Decisión[24] son los siguientes:

1. **Listar todas las causas probables dentro de la Tabla de Decisión:** Se deben de anotar todas las causas y los valores posibles.

Causas	*Valores*								
c_1	$SI \vee NO$								
c_2	$SI \vee NO$								
c_3	$SI \vee NO$								
Acciones									

   Para este caso vemos que están presentes tres causas con 2 valores posibles por cada una.

2. **Calcular el número de combinaciones posibles:** Esto se calcula tomando primero el número de valores posibles como potencia del número de causas con estos valores. Esto es si todas las causas tienen dos valores simples $SI \vee NO$ entonces las combinaciones posibles serían, dados n números de valores y m número de causas tenemos n^m combinaciones posibles, en

cambio si tenemos una causa con tres valores y tres causas con dos valores tendríamos $3^1 \times 2^3$ = 24 combinaciones posibles.

Combinaciones

Causas	*Valores*	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
c_1	$SI \vee NO$								
c_2	$SI \vee NO$								
c_3	$SI \vee NO$								
Acciones									

Se puede observar en la tabla de decisión anterior que como tenemos tres causas con 2 valores para cada una, el cálculo de las combinaciones seria igual a 2^3 = 8.

3. **Llenar las columnas:** Para poder llevar a cabo el proceso de llenado de las columnas se establecen los siguientes tres pasos:

   a) Se debe determinar cual va a ser el Factor de Repetición (*Repeating Factor (RF)*) el cual se calcula al dividir las combinaciones restantes por el número de valores posibles para esa causa.

   b) Se deben de escribir los tiempos RF del primer valor presente, después los del siguiente y así sucesivamente, hasta que una fila esta llena por completo.

   c) Se pasa a la siguiente fila y se regresa al paso 1.

Combinaciones

Causas	*Valores*	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
c_1	$SI \vee NO$	SI	SI	SI	SI	NO	NO	NO	NO
c_2	$SI \vee NO$	SI	SI	NO	NO	SI	SI	NO	NO
c_3	$SI \vee NO$	SI	NO	SI	NO	SI	NO	SI	NO
Acciones									
a_1									
a_2									

4. **Adicionar acciones a la tabla de decisión:** Se necesita leer columna por columna y determinar cuales son las acciones. Un efecto puede ocurrir en combinaciones múltiples de prueba.

Combinaciones

Causas	*Valores*	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
c_1	$SI \vee NO$	SI	SI	SI	SI	NO	NO	NO	NO
c_2	$SI \vee NO$	SI	SI	NO	NO	SI	SI	NO	NO
c_3	$SI \vee NO$	SI	NO	SI	NO	SI	NO	SI	NO
Acciones									
a_1		$\times$		$\times$	$\times$			$\times$	$\times$
a_2			$\times$			$\times$	$\times$		$\times$

5. **Reducir el número de combinaciones:** Se deben encontrar condiciones indiferentes, estas se presentan cuando a pesar de tener diferentes combinaciones de valores para cada causa presente en el problema, generan la misma acción. Una vez que se localizan dichas condiciones indiferentes, unimos las columnas en una sola y ponemos un '-' en donde difieran las columnas que se juntan.

Combinaciones

Causas	*Valores*	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
c_1	$SI \vee NO$	SI	SI	SI	SI	NO	NO	NO	NO
c_2	$SI \vee NO$	SI	SI	NO	NO	SI	SI	NO	NO
c_3	$SI \vee NO$	SI	NO	SI	NO	SI	NO	SI	NO
Acciones									
a_1		$\times$		$\times$	$\times$			$\times$	$\times$
a_2			$\times$			$\times$	$\times$		$\times$

Combinaciones

Causas	*Valores*	v_1	v_2	v_3	v_5	v_6	v_7	v_8
c_1	$SI \vee NO$	SI	SI	SI	NO	NO	NO	NO
c_2	$SI \vee NO$	SI	SI	NO	SI	SI	NO	NO
c_3	$SI \vee NO$	SI	NO	$-$	SI	NO	SI	NO
Acciones								
a_1		$\times$		$\times$			$\times$	$\times$
a_2			$\times$		$\times$	$\times$		$\times$

Podemos observar en la tabla de decisión que la combinación v_3 y v_4 presentan diferentes valores para la causa 3, pero producen la misma acción, esta es una condición indiferente por lo que reducimos estas dos combinaciones en una sola y ponemos un '-' en el valor que difieren.

6. **Eliminación de combinaciones imposibles:** Se eliminan de la tabla de decisión obtenida hasta el paso 5, aquellas condiciones que no pueden ser llevadas acabo por alguna contradicción de la especificación del problema a resolver, en esta tabla de decisión que se esta construyendo no podemos especificar este paso, pero más adelante en el ejemplo lo revisaremos mas a detalle para dejar claro la realización de este paso. Sin embargo, supongamos que con la tabla de decisión que se ha venido planteando, que se eliminan las combinaciones v_6, v_7 y v_8, por lo que obtenemos la siguiente tabla de decisión aplicando todos los pasos necesarios.

		Combinaciones			
Causas	*Valores*	v_1	v_2	v_3	v_5
c_1	$SI \vee NO$	SI	SI	SI	NO
c_2	$SI \vee NO$	SI	SI	NO	SI
c_3	$SI \vee NO$	SI	NO	$-$	SI
Acciones					
a_1		$\times$		$\times$	
a_2			$\times$		$\times$

2.10.4. Ejemplo de uso de Tablas de Decisión

Plantearemos el siguiente ejemplo de uso de las Tablas de Decisión:

> *Una tienda de ropa saca la siguiente promoción, se otorga un descuento del 5 % si la compra es mayor de $500 y del 15 % si es mayor de $2000. Para los clientes que compran con frecuencia se hace una excepción y se les da por lo menos el 5 % de descuento*

Primero enumeramos las causas posibles:

1. *Compra Pequeña* [1, 500]
2. *Compra Mediana* [501, 2000]
3. *Compra Grande* >1000
4. *Cliente frecuente*

Después enumeramos las acciones posibles:

1. Descuento = 0
2. Descuento = 5 %
3. Descuento = 15 %

Una vez que se tienen estos componentes se determina el número total de reglas posibles que serán la cantidad de columnas en la Tabla de Decisión.

- *Compra Pequeña* tiene 2 valores ($SI \vee NO$)
- *Compra Mediana* tiene 2 valores ($SI \vee NO$)
- *Compra Grande* tiene 2 valores ($SI \vee NO$)

- *Cliente frecuente* tiene 2 valores ($SI \vee NO$)

Total alternativas = $2 \times 2 \times 2 \times 2 = 16$ reglas posibles

Ya podemos diseñar cual va ser la estructura de la Tabla de Decisión considerando todas las combinaciones posibles:

		Combinaciones									
Causas	*Valores*	v_1	v_2	v_3	v_4	v_5	...	v_{13}	v_{14}	v_{15}	v_{16}
Compra Pequeña	$SI \vee NO$						...				
Compra Mediana	$SI \vee NO$						...				
Compra Grande	$SI \vee NO$						...				
Cliente frecuente	$SI \vee NO$						...				
Acciones											
Descuento = 0							...				
Descuento = 5							...				
Descuento = 15							...				

En esta tabla de decisión tenemos presentes 4 causas posibles con 2 valores $SI \vee NO$, por lo que tenemos presentes 16 combinaciones posibles y ademas tenemos 3 acciones que se deben de realizar, estas son las características que conforman esta tabla de decisión.

Ahora podemos empezar llenando todas las combinaciones posibles con los valores que tiene cada causa, empezamos con la causa *Cliente frecuente*, alternando los valores $SI \vee NO$, de uno en uno, después seguimos con la causa *Compra Mediana* alternándolos de acuerdo con los grupos de valores que fueron fijados para la anterior condición, de tal forma que en la tabla existan todas combinaciones posibles. Esto se repite hasta que todas las condiciones presentes quedan llenadas como se muestra en siguiente en la tabla de decisión que estamos formando para este ejemplo:

		Combinaciones									
Causas	*Valores*	v_1	v_2	v_3	v_4	v_5	...	v_{13}	v_{14}	v_{15}	v_{16}
Compra Pequeña	$SI \vee NO$	SI	SI	SI	SI	SI	...	NO	NO	NO	NO
Compra Mediana	$SI \vee NO$	SI	SI	SI	SI	NO	...		NO	NO	NO
Compra Grande	$SI \vee NO$	SI	SI	NO	NO	SI	...	SI	SI	NO	NO
Cliente frecuente	$SI \vee NO$	SI	NO	SI	NO	SI	...	SI	NO	SI	NO
Acciones											
Descuento = 0											
Descuento = 5											
Descuento = 15											

Procedemos ahora a identificar los casos imposibles y se eliminan de la Tabla de Decisión, para que una vez que estos han sido eliminados se puedan poner las acciones correspondientes a cada combinación:

		Combinaciones									
Causas	*Valores*	v_1	v_2	v_3	v_4	v_5	...	v_{13}	v_{14}	v_{15}	v_{16}
Compra Pequeña	$SI \vee NO$	SI	SI	SI	SI	SI	...	NO	NO	NO	NO
Compra Mediana	$SI \vee NO$	SI	SI	SI	SI	NO	...		NO	NO	NO
Compra Grande	$SI \vee NO$	SI	SI	NO	NO	SI	...	SI	SI	NO	NO
Cliente frecuente	$SI \vee NO$	SI	NO	SI	NO	SI	...	SI	NO	SI	NO
Acciones											
Descuento = 0											
Descuento = 5											
Descuento = 15											

		Combinaciones					
Causas	*Valores*	v_7	v_8	v_{11}	v_{12}	v_{13}	v_{14}
Compra Pequeña	$SI \vee NO$	SI	SI	NO	NO	NO	NO
Compra Mediana	$SI \vee NO$	NO	NO	SI	SI	NO	NO
Compra Grande	$SI \vee NO$	NO	NO	NO	NO	SI	SI
Cliente frecuente	$SI \vee NO$	SI	NO	SI	NO	SI	NO
Acciones							
Descuento = 0			$\times$				
Descuento = 5		$\times$			$\times$		
Descuento = 15				$\times$		$\times$	$\times$

Estos casos imposibles resultan de que consideramos todos las posibles combinaciones que se pueden presentar, en donde algunas de ellas no cumplen las especificaciones del problema que estemos resolviendo, podemos ver que el caso v_1, v_2, ..., v_5, se consideraron imposibles ya que una compra no puede ser de más de un tipo porque los intervalos de compra fueron establecidos desde el principio, una compra pequeña va de \$1 a \$500 y no puede existir un caso en el que sea una compra pequeña y a la vez una compra mediana o grande, es por ello que estos casos son eliminados de la tabla de decisión.

Se analiza la tabla para determinar si existen causas indiferentes y se marcan con un '-', una vez identificadas se juntan en una sola columna para reducir la tabla, este es el ultimo paso que se lleva acabo para obtener la tabla de decisión final para este ejemplo, de la siguiente forma:

Combinaciones

Causas	*Valores*	v_7	v_8	v_{11}	v_{12}	v_{13}	v_{14}
Compra Pequeña	$SI \vee NO$	SI	SI	NO	NO	NO	NO
Compra Mediana	$SI \vee NO$	NO	NO	SI	SI	NO	NO
Compra Grande	$SI \vee NO$	NO	NO	NO	NO	SI	SI
Cliente frecuente	$SI \vee NO$	SI	NO	SI	NO	SI	NO
Acciones							
Descuento = 0			$\times$				
Descuento = 5		$\times$			$\times$		
Descuento = 15				$\times$		$\times$	$\times$

$\Downarrow$

Combinaciones

Causas	*Valores*	v_7	v_8	v_{11}	v_{12}	v_{13}
Compra Pequeña	$SI \vee NO$	SI	SI	NO	NO	NO
Compra Mediana	$SI \vee NO$	NO	NO	SI	SI	NO
Compra Grande	$SI \vee NO$	NO	NO	NO	NO	SI
Cliente frecuente	$SI \vee NO$	SI	NO	SI	NO	SI
Acciones						
Descuento = 0			$\times$			
Descuento = 5		$\times$			$\times$	
Descuento = 15				$\times$		$\times$

2.10.5. Ventajas y desventajas de Reglas de Decisión

- Con frecuencia describir un problema mediante palabras deja mucha confusión en los términos utilizados, ya que no se puede establecer con claridad los objetivos que se buscan, es por esto que las Tablas de Decisión son de gran ayuda ya que nos permiten manejar de una forma más clara y sencilla las diferentes situaciones que se pueden presentar para un

problema dado, así como dar las solución o las acciones recomendadas para cada una de esta combinaciones

- Las Tablas de Decisión suelen verse como procesos de decisión en ambiente de riesgo, caracterizado por tener asociado una probabilidad de ocurrencia a cada combinación de estados. Estas probabilidades pueden ser conocidas o estimadas previamente antes del proceso de toma de decisiones.

- Este tipo de modelos pueden ser usadas en diferentes áreas de investigación como: Análisis de Negocios, Programación, Pruebas de negocio, Diseño de hardware, etc.

- Para problemas muy grandes en las que se presentan muchos casos, estas tablas crecen de manera exponencial y el tiempo de construir la tabla suele incrementarse por ello se debe de buscar formas de como poder reducir el tamaño de las Tablas de Decisión para su aplicación a problemas complicados.

Capítulo 3

Extracción de conocimiento en Bases de Datos.

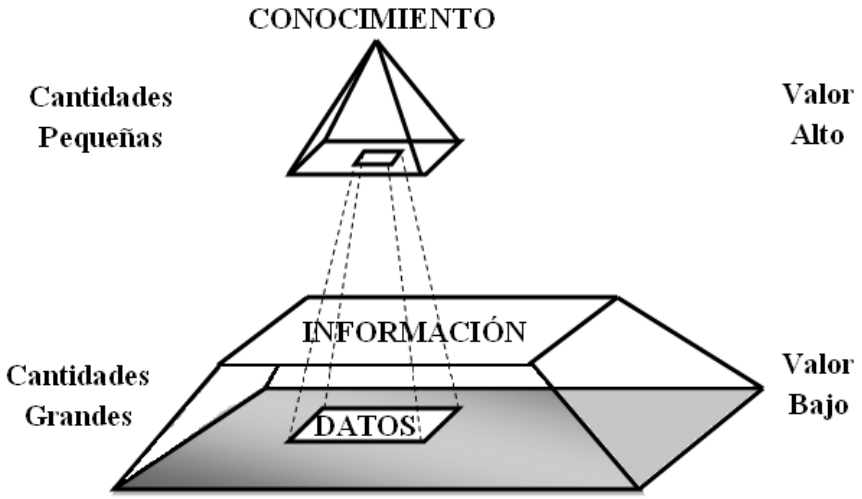
3.1. Introducción

La extracción de información útil (conocimiento), sobre BD grandes ha originado una nueva generación de Teorías de Cómputo y de herramientas que permitan poder explorar, analizar y finalmente obtener dicho conocimiento, estas teorías son aplicadas en el proceso llamado Extracción de Conocimiento en Bases de Datos[7] (*Knowledge Discovery in Databases* - KDD).

El incremento de las BD en los últimos diez años, toma un papel muy importante para que diera origen a este proceso, ya que a mayor cantidad de datos que generan las organizaciones, mayor es su desconocimiento de toda esta gran cantidad de datos que solo representa la materia prima bruta.

Cuando estas grandes cantidades de datos representan un significado especial para el desarrollo de las organizaciones, son consideradas *información*. En el momento en que los datos pasan a ser información, especialistas en el manejo de BD grandes, dan una interpretación a esta información mediante la construcción de un modelo que represente un valor agregado a las organizaciones, se dice entonces que se obtiene un *conocimiento* real de aquellas grandes cantidades de datos. En la Figura 3.1 se muestra un diagrama de la jerarquía que existe entre las principales partes que conforman una BD (datos, información y conocimiento).

Figura 3.1: Esquema de la jerarquía de los componentes en una BD.



El objetivo principal de KDD, es el desarrollo de métodos y técnicas para encontrar dicho conocimiento en BD que se consideran del mundo real, estas BD caen dentro de la clasificación cuatro, cinco y seis de la Tabla 1 presentada en la Sección 1 de este trabajo.

Los datos presentan un valor real cuando de ellos se obtiene un conocimiento, que puede ayudarnos a tomar mejores decisiones a una serie de problemas. El problema básico que aborda el KDD es el de convertir los datos en información, para convertirlas en formas más compactas (por ejemplo, un informe corto), más abstractas (por ejemplo, una aproximación o un modelo descriptivo del proceso que generó los datos), o más útiles (por ejemplo, un modelo predictivo para estimar el valor de los casos futuros).

Las metas principales de KDD son las siguientes:

- Se debe de procesar de manera automática, grandes cantidades de datos crudos (este término se utiliza cuando llamamos a los materiales en su forma natural, como por ejemplo el petroleo crudo).

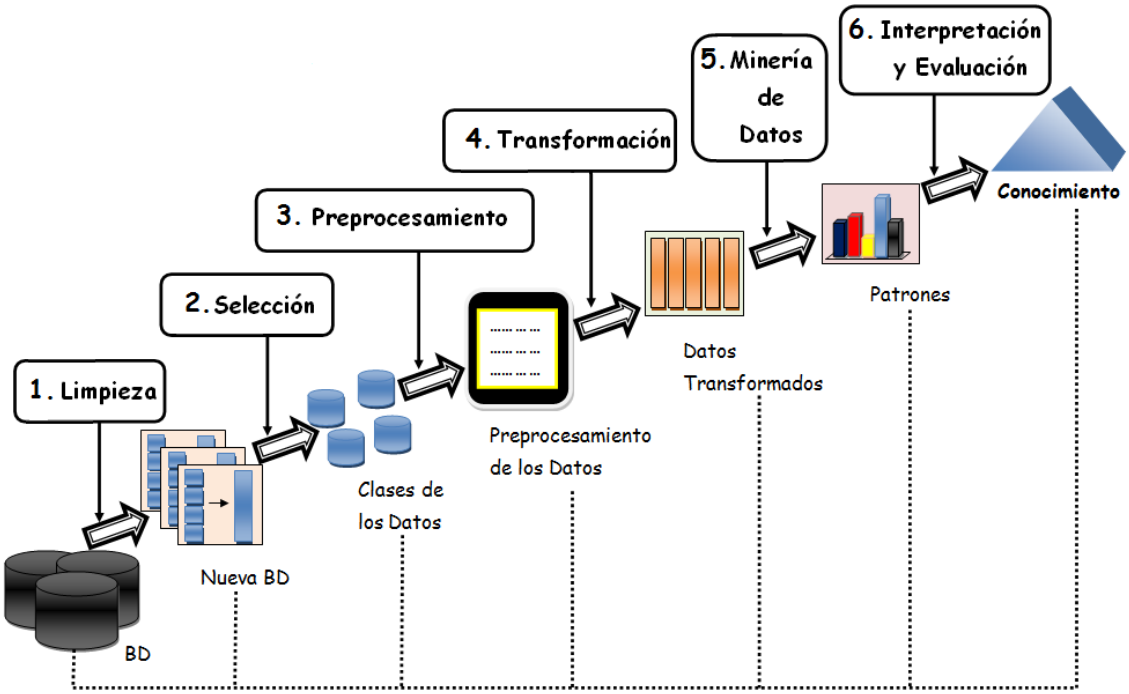
- Dentro de estas grandes cantidades de datos debemos de identificar cuales van a ser los patrones más significativos y relevantes.

- Una vez que se tiene identificados a estos, se les da una interpretación para poderlos presentar como conocimiento útil, para poder resolver la problemática que se plantea al inicio de cualquier proyecto que tenga implícito el uso de KDD.

3.2. El proceso de KDD

Para poder llevar a cabo KDD se debe de tener una guía de como poder realizar la obtención del conocimiento en BD grandes, en donde se especifique los pasos que deben ser realizar al momento de tener en nuestro poder grandes cantidades de datos, a esta guía se le conoce como el *Proceso de KDD*. Veremos que una de las fases más importantes en el proceso de KDD es la de Minería de Datos, ya que nos permite aplicar métodos para poder encontrar patrones dentro de los datos, los cuales posteriormente son convertidos en conocimiento. El diagrama de la Figura 3.2 muestra dicho proceso con cada una de las fases que lo componen:

Figura 3.2: Proceso del KDD.



Cada una de estas fases consiste en lo siguiente:

- **Limpieza:** Al iniciar el proceso de KDD, usualmente se cuenta con una BD, que contiene todo lo almacenado por una organización sobre algún contexto. Podemos pensar en una BD que nos ofrece algún banco, con la información de todos sus clientes que tienen una tarjeta de crédito, este sería el contexto de la BD.

  En ella se tiene las características esenciales de los clientes, como identificador del cliente, nombre, apellidos, edad, RFC(Registro Federal de Contribuyentes), domicilio, teléfono, empresa donde labora, ingresos, etc, cada una de estas características del cliente se le denominan atributos. Cada uno de los atributos tiene un conjunto de valores que distinguen si un atributo es *discreto* en caso que se tenga un número finito o contable de valores o *contínuo* si tiene un número infinito de valores posibles.

  Cada uno de estos atributos presenta un diferente grado de beneficio para el contexto que se esta manejando, así también algunos de los valores asignados para cada ejemplo en la BD puede presentar errores o inconsistencias. Esta fase dentro del proceso de KDD es la encargada de tratar con este estado inicial de los datos en las BD, tenemos los objetivos de eliminación de ruido e inconsistencias sobre los datos, encontrar los atributos que nos aporten mayor beneficio al contexto y la combinación de múltiples fuentes de datos.

- **Selección:** Una vez que tenemos una BD más refinada, gracias a la fase de limpieza, es aplicada la fase de selección, para poder encontrar los datos relevantes para el problema, las clases que distinguen cada uno de los ejemplos. En dado caso que no existan dichas clases debemos de aplicar técnicas de agrupamiento para asignarla algún tipo de etiqueta a las clases y finalmente la unión de varias fuentes de datos que puedan enriquecer el tamaño de los datos a ser utilizados, El estado que tiene la BD es más completa que la que se obtiene en la fase de limpieza, ya que se incluyen las clases que van ser de suma importancia al llegar a la fase de Minería de Datos.

- **Preprocesamiento y Transformación:** Una vez que los datos han sido pasados por la fase de limpieza y selección, los datos aun tienen que tener un formato para poder aplicar los métodos

de Minería de Datos, es donde las fases de preprocesamiento y transformación son aplicadas. Dentro de las herramientas que nos permite aplicar Minería de Datos sobre BD grandes, se utiliza un formato especial para el manejo de las BD, algunas manejan el formato relacional de SQL o DB2 por ejemplo, mientras que otras tienen su propio formato, es aquí donde nosotros debemos de decidir primero qué herramientas vamos a usar para aplicar Minería de Datos, para que así sepamos de antemano qué formato debe llevar la(s) BD que ocupemos para aplicar el proceso de KDD. Una vez que se tiene establecido cual herramienta vamos a usar debemos de aplicar la fase de procesamiento y transformación para dejar en un formato diferente a la BD que será usada.

- **Minería de Datos:** Al llegar a esta fase del procesos de KDD, se considera que el proceso realmente ha empezado, ya que la parte medular se encuentra en la fase de Minería de Datos. Es aquí donde se aplican métodos de extracción de patrones y son calculadas medidas de funcionamiento para evaluar el rendimiento de estos métodos. En este trabajo presentaremos dos formas de realizar esta fase, la de escoger a una serie de herramientas para llevar a cabo Minería de Datos y también la forma en la que nosotros implementamos dichos métodos y calculamos las medidas de rendimiento. En esta fase se plantean una serie de metas que se pueden realizar al aplicar Minería de Datos las cuales son:

  - **Predicción:** Descubrir la manera de cómo ciertos atributos, dentro de los datos, se comportarán en el futuro.
  - **Identificación:** Identificar la existencia de objetos, eventos y actividades dentro de los datos.
  - **Clasificación:** Particionar los datos de acuerdo a las clases o etiquetas que sean asignadas a cada ejemplo en los datos.
  - **Agrupamiento:** Nos permite maximizar las similaridades y minimizar las diferencias entre los objetos, mediante algún criterio de agrupamiento.
  - **Asociación:** Las reglas de asociación intentan descubrir cuales son las conexiones que se pueden tener entre los objetos identificados.

- **Interpretación y Evaluación:** Se realiza un análisis de los patrones obtenidos hasta la fase de Minería de Datos, mediante técnicas de visualización y de representación del conocimiento, para obtener la solución a la problemática implícita dentro de la BD, el objetivo principal que es encontrar el conocimiento que de un valor más significativo a los datos.

Para este trabajo mostraremos como fue usado el proceso de KDD a una BD del mundo real, en donde observaremos como se aplicó cada una de las fases de KDD en esta BD, es por ello que debemos de presentar cuál es el formato que tiene esta BD y la problemática que se desea resolver.

3.3. Forma de la BD utilizada.

La fuente principal que tuvimos para este trabajo fue la de ECML PKDD (The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases[10]), este es uno de los Congresos más reconocidos para el área de Aprendizaje Computacional y Minería de Datos, realizado en Europa cada año. En su haber lleva 19 ECML y 12 conferencias de PKDD y en cada conferencia se proporciona una BD, de la cual se realiza un reto de extraer conocimiento.

Esta BD fue proporcionada por por la Empresa Gemius[16], dedicada al monitoreo de Internet sobre Europa central y oriental, a la EMCL PKDD 2007[10]. Esta BD representa un reto, ya que se encuentra entre la clasificación 4 y 5 de la Tabla 1.1 vista en el Capítulo 1, esto justifica que podamos aplicar Minería de Datos ya que debemos encontrar el conocimiento que se encuentra implícito en esta BD.

Algunos de los conceptos más importantes que describen el contexto de esta BD son los siguientes:

- Una *visita* se define como una secuencia de páginas vistas por un usuario, *cookies,* a lo largo de un determinado tiempo de monitoreo.

- En cada *visita* existen diferentes *páginas Web* que son identificadas por sus *categorías*.

- Durante la visita de un usuario, se pueden ver las páginas de una o más categorías. Por lo tanto se proponen las siguientes dos clases de visita:

  - *Visita Corta:* Es una visita con páginas consultadas de una sola categoría.
  - *Visita Larga:* Es una visita en donde se revisaron dos o más categorías.

De esta BD ocupamos los siguientes dos archivos:

- **Gemius_users:** Este archivo contiene los atributos que describen las características de los usuarios que fueron monitoreados por la Empresa Gemius. Este archivo contiene 4882 ejemplos diferentes y en cada uno de estos hay 8 atributos o campos diferentes, los cuales presentan un número finito de valores por lo cual son atributos discretos. La estructura de este archivo se muestra en la Tabla 3.1.

Tabla 3.1: Tabla de Usuarios.

id_usuario	id_país	id_región	id_cuidad	id_sistema	id_subsistema	id_navegador	id_ver_navegador
...	...	...	...	...	...	...	...
10	42	11	44	3	9	1	517

Cada uno de los atributos en esta tabla se describen a continuación:

1. *id_usuario*: Identificador web de los usuarios (basado en el registro de los cookies del usuario).
2. *id_país, id_región, id_ciudad*: Estos tres atributos nos proporcionan la geolocalización de los datos derivados de las direcciones IP de los usuarios.
3. *id_sistema, id_subsistema*: Son los identificadores de los Sistemas Operativos y sus versiones de los usuarios.
4. *id_navegador, id_ver_navegador*: Son los identificadores de los Navegadores y sus versiones de los usuarios.

- **Gemius_paths_training_data:** Este archivo es el conjunto de entrenamiento proporcionado por la Empresa Gemius, en el cual se encuentran, qué tipo de visitas fueron realizadas por los usuarios seleccionados para el monitoreo. En este archivo contaba con 379485 ejemplos, con 4 atributos distintos cada uno de ellos, la estructura de esta segunda tabla se muestran en la Tabla 3.2:

Tabla 3.2: Tabla de Caminos de Visita.

id_camino	id_usuario	tiempo_visita	camino (id_categoria, número_paginas_visitadas)..., ..., ...				
...	...	...	...				
27	1	1169814548	7,1	16,2	17,9	16,1	

Cada uno de los atributos en esta tabla se describen a continuación:

1. *id_camino*: Es el identificador de cada camino de visita.
2. *id_usuario:* Este campo identifica que usuario realizó la visita.
3. *tiempo_visita*: Define el tiempo en el que se empezó la consulta de la visita por un usuario.
4. *id_categoría*: Es el identificador de la categoría de la página web visitada por algún usuario.
5. *paginas_visitadas*: Este atributo nos proporciona el número de páginas vistas durante una vista (esta se interrumpe cuando el usuario escoge una categoría distinta)

Esta BD presentaba una serie de problemas a resolver, las cuales fueron el reto para la EMCL PKDD 2007, de este conjunto de problemas nos enfocaremos en este trabajo a resolver uno de ellos el cual es el siguiente:

- **La longitud de la vista.** Dado cualquier ejemplo de prueba, se tenía que determinar la clase de visita a la que pertenece.

En las siguientes secciones mostraremos como aplicamos cada una de las fases del proceso de KDD para esta BD, hasta llegar a la fase de Minería de Datos en la cual se realizaron un conjunto de pruebas y resultados que incluyen el uso de un grupo de clasificadores de Aprendizaje Automático y la aplicación de nuestra propuesta desarrollada.

3.4. Aplicación de la Fase de Limpieza.

La BD utilizada en un principio se encontraba por separado en los dos archivos mostrados en la sección anterior, es por esto que empezamos a trabajar por separado con los dos archivos.

El primer archivo que se reviso fue el de `Gemius_users`, en este pudimos observar que su estructura era la de una tabla, en donde cada uno de los atributos de un ejemplo en particular estaba separado por un espacio en blanco hasta llegar con el último atributo, formando la Tabla de Usuarios. El formato de esta tabla no presentó ningún problema ya que cada ejemplo esta perfectamente construido y los espacios en blanco los podíamos quitar y sustituir por otro cosa como una coma, un punto o un punto y coma según fuera el formato para las BD usadas por las herramienta de Minería de Datos seleccionadas. La única modificación que realizamos a esta tabla en la fase de limpieza fue quitar aquellos atributos que tenían la función de identificadores ya que no aportaban ningún beneficio.

Para el segundo archivo `Gemius_paths_training_data`, presentaba la misma estructura de una tabla separada por espacios en blanco para distinguir cada uno de los atributos presentes, es en esta tabla donde se localizó un primer problema. Este se encontraba en el atributo de caminos de visita, este atributo describía el camino de alguna visita por alguno de los usuarios y tenía implícitos dos subatributos, id_categoría y paginas_visitadas lo que presentaba el siguiente formato:

id_camino	*Camino (id_categoría, paginas_visitadas) (. . .) , (. . .)*
248	18,2 9,2
249	12,10
250	11,1 7,13 12,5 9,1
. . .	. . .
273	12,5 17,3 12,2 17,3 12,3

Esto fue considerado un primer problema a resolver en la fase de limpieza, ya que el formato de las BD para las herramientas de Minería de Datos consideradas, no manejaban este tipo de atributos que contribuyeran duplas como los presentes en el atributo caminos de visita. Se procedió entonces a quitar las comas por espacios en blanco, esto es si algunos de los ejemplos presentaba la dupla 18,2, la coma era eliminada y se procedía a separar esta dupla por un espacio en blanco teniendo ahora '18 2', esta primera aproximación para solucionar este conflicto no era valida, a veces el tamaño de las visitas variaba de una categoría visitada a más de cien, esto implicaba que debamos localizar el número mayor de categorías visitadas, para agregar a cada uno de los registros con menor número de categorías valores nulos para que la longitud de los atributos fuera el mismo para cada ejemplo presente en la BD. Esto implicaba un incremento en el número de atributos que no aportaban ningún beneficio, algo que en la fase de limpieza se evita, por lo que esta solución fue descartada.

La segunda opción que realizamos para dale solución a este problema, fue la de juntar las categorías visitadas en un solo atributo, separando en caso de que existieran más de una por un cero, para el número de páginas visitadas se agregaba otro atributo de la misma forma. Lo que pensábamos realizar es lo siguiente:

id_camino	*id_categoría*	*paginas_visitadas*
248	1809	202
249	12	10
250	110701209	10130501
. . .	. . .	. . .
273	12017012017012	503020303

Esta opción al principio presentaba una buena elección de como poder acomodar los datos, pero se presento el inconveniente de encontrar una forma automática de hacerlo. Al no encontrar dicha forma de realizarlo tuvimos que hacerlo de forma manual lo que nos demoró mucho tiempo, por lo que consideramos que la perdida de tiempo en tratar de poner todas los registros que eran un total de 379485 registros era demasiada para los propósitos del trabajo, es por ello que descartamos también esta opción.

Como cada registro de la Tabla de caminos de visita tenía una o más categorías visitadas, se cambio la forma de la tabla, incrementando el número de registros, esto es, si un registro presentaba tres categorías visitadas, la nueva estructura de la tabla sería, con tres registros, en vez de uno y cada uno de ellos tendría un atributo más que llamamos orden de visita, para entender esto pensemos en que se tiene un registro con 5 categorías, a continuación se muestra como sería su cambio con este nuevo formato de la tabla:

id_camino	*Camino (id_categoría, paginas_visitadas) (. . .) , (. . .)*
273	12,5 17,3 12,2 17,3 12,3

id_camino	*id_categoría*	*paginas_visitadas*	*orden*
273	12	5	1
273	17	3	2
273	12	2	3
273	17	3	4
273	12	3	5

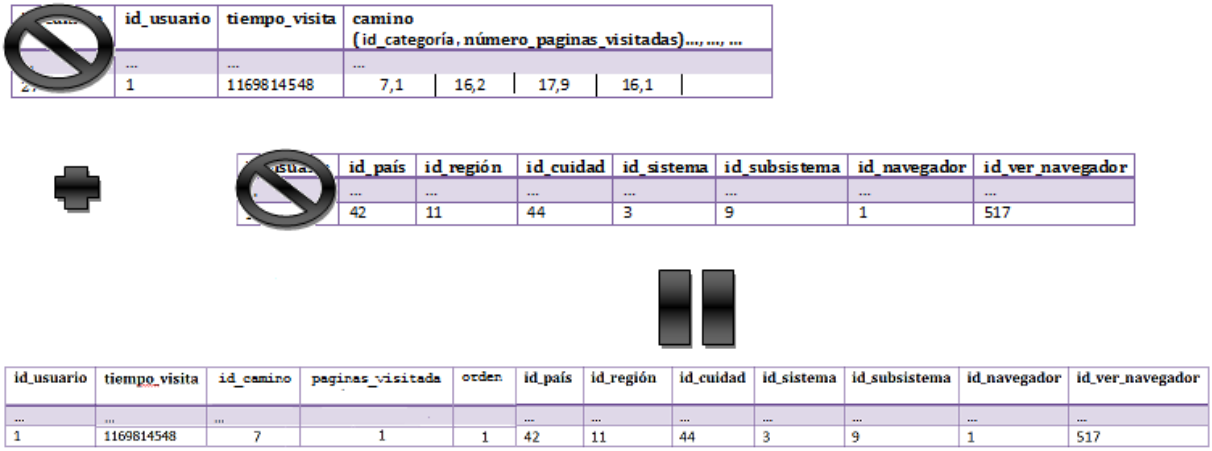
Podemos ver en este ejemplo se tenía al principio un registro con el id 273, en el cual para el atributo *Camino*, 2 categorías distintas visitadas por el usuario, en este caso las categorías presentes son la 12 y la 17, presentando diferentes páginas visitadas en cada caso, lo que se realiza para modificar este ejemplo es crear 5 registros nuevos con diferente estructura en el cual se agrega tres atributos más, el *id_categoría* que nos dice qué categoría fue visitada por el usuario, *paginas_visitadas* en donde tenemos cuantas páginas fueron visitadas por cada categoría y el atributo *orden*, que nos dice en que orden fueron visitadas las distintas categorías, en caso de existir más de una. Una vez

que completamos la modificación para cada registro, eliminamos el atributo *id_camino*, ya que es solo un identificador de esta tabla y no representa mayor beneficio. Esta es la forma que escogimos para aplicar la fase de limpieza de la BD creando dos tablas diferentes de como las que teníamos al principio.

3.5. Aplicación de la Fase de Selección.

Para esta fase identificamos cuales eran los atributos que nos proporcionaban mayor beneficio, algo que realizamos desde la fase anterior, por lo que para esta fase de selección juntamos las dos tablas, para formar una con los atributos que resultaron al aplicar la fase de limpieza, esta nueva tabla la llamamos gemius_completa, la cual es la BD que haremos referencia a lo largo de este trabajo, la cual obtuvimos juntando las dos fuentes de datos que teníamos al inicio, la estructura de esta se puede ver en la Figura 3.3.

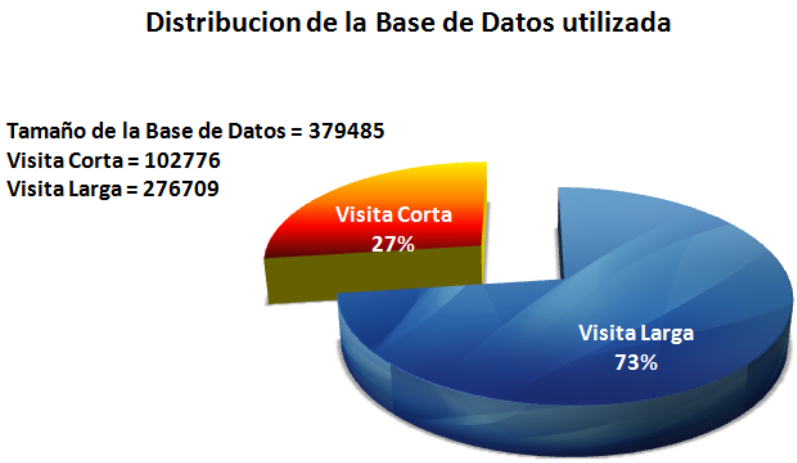
Figura 3.3: Tabla gemius_completa.



Una vez que se obtuvo esta BD, se procedió a identificar las clases de cada uno de los registros o ejemplos, en este caso se cuenta con la clase *Visita Corta* y *Visita Larga*, que diferenciaban a cada uno de los ejemplos, esta BD cumple con las características del aprendizaje supervisado que vimos en la Sección 2.1. La distribución de la BD gemius_completa con las dos clases presentes se

muestra en la Figura 3.4.

Figura 3.4: Distribución de la BD gemius_completa.



3.6. Aplicación de la Fase de Preprocesamiento y Transformación.

El preprocesamiento de esta base de datos consistió en decidir qué tipo de valores debíamos considerar, para todos los atributos excepto el de la clase dejamos los valores tal y como estaban, ya que todos eran atributos discretos, para el caso de las clases se cambiaron tomando las siguientes clases: *Visita Corta* = 1 y *Visita Larga* = 2.

Para la fase de transformación, estuvo en dependencia de las herramientas de Minería de Datos que seleccionamos, a manera de adelanto listaremos las cuatro que seleccionamos aunque no entraremos a explicar a detalle en el funcionamiento de cada una de ellas solo explicaremos como tuvimos que acomodar la BD para hacer uso de estas:

- TANAGRA (A free data mining software for research and education). El creador de este software es Ricco Rakotomalala en Lyon Francia, es un software libre para propósitos académicos y de investigación para la área de Minería de Datos. Se tienen diferentes métodos de

Minería de Datos para la exploración de BD grandes.

- SIPINA (Supervised Learning). Al igual que TANAGRA fue desarrollado por Ricco Rakotomalala, es un software el cual contiene un gran número de implementaciones de métodos de Árboles de Decisión y otros tipos de clasificadores, para poder llevar a cabo la clasificación de datos. Es distribuido desde el año de 1995 de forma gratuita.

- WEKA (Waikato Environment for Knowledga Analysis). Este es un software desarrollado por la Universidad de Waikato[36], escrito en Java, y distribuido como un software libre distribuido bajo licencia GNU-GPL, para poder realizar prácticas de Aprendizaje Computacional y Minería de Datos.

- Implementaciones propias realizadas en Matlab R2008b (7.7).

Para las tres herramientas se utiliza un formato ARFF (*attribute-relation file format*) utilizado por WEKA principalmente, al igual que en SIPINA y TANAGRA, es por ello que a la BD le dimos este formato, el cual separa cada uno de los atributos por una coma, por lo que los espacios en blanco que originalmente tenía la BD los sustituimos por una coma, además agregamos la siguientes líneas en la cabecera de la BD:

@relation gemius_completa

@attribute id_usuario numeric

@attribute id_país numeric

@attribute id_región numeric

@attribute id_cuidad numeric

@attribute ⋮ numeric

@attribute class 1,2

@data

En la cabecera del formato ARFF tenemos al inicio el nombre de la relación de la BD, después se pone cada uno de los atributos con su tipo de valor en este caso todos son enteros (*numeric*), al final de este encabezado ponemos las clases de la BD, las cuales han sido cambiada por 1 y 2. Después tenemos cada uno de los ejemplos con las clases a las que pertenece cada uno de los diferentes ejemplos. Para las tres primeras herramientas usamos esta transformación, ya que para las implementaciones que realizamos en Matlab 7.7 pueden usar el formato inicial de la BD que tiene separada a cada uno de los atributos por un espacio en blanco, poniéndolos en una arreglo de tipo matriz. Las herramientas WEKA, TANAGRA y SIPINA seran vistos a detalle en la Sección 5.1.1, Sección 5.1.2 y Sección 5.1.3.

3.7. Aplicación de la Fase de Minería de Datos, Interpretación y Evaluación.

Como se vió en la fase de preprocesamiento y transformación, mecionamos las 4 diferentes herramientas para Minería de Datos que se van a utilizar. En este trabajo presentaremos como se resolvió el problema presentados en la Sección 3.3, en donde se habló acerca de la forma de la BD, este es un problema de clasificación, es por esto que utilizamos aquellos métodos de clasificación que nos proporcionan cada una de estas herramientas, así como algunas implementaciones desarrolladas. También se aplicó el algoritmo de Mezcla Genética de Expertos Ponderada (WGME) que fue la propuesta realizada para este trabajo que explicaremos a detalle en el Capítulo 4. En la fase de Minería de Datos realizamos una serie de experimentos para darle solución al problema de la longitud de la vista. Se seleccionaron diferentes herramientas para aplicar la clasificación sobre la BD gemius_complete, tomando diferentes conjuntos de prueba y aplicando una serie de clasificadores y obteniendo el rendimiento de cada uno de ellos. Para poder evaluar con alguna prueba estadística estos resultados, estas pruebas son aplicadas para realizar la fase de interpretación y evaluación, en donde seleccionamos la prueba llamada t-Test que será descrita a continuación.

3.7.1. t-Test

Este método estadístico se utiliza para saber cual es el nivel de significancia entre dos conjuntos de muestras[19]. En este caso, las muestras serán los valores de alguna de las medidas de rendimiento que ocupemos, en este caso la exactitud que obtengamos con los clasificadores utilizados. Este método estadístico realizada a un conjunto de muestras encontradas al realizar la tarea de clasificación será revisada en el Capítulo 5, por lo que en este momento detallaremos los pasos que se deben de realizar para el método t-Test, para un conjunto de ejemplo, en donde consideramos dos clasificadores con distintos valores aleatorios de exactitud y para un total de 15 muestras.

1. Si tenemos un número total de muestras n, calculando la diferencia entre cada uno de los resultados para cada clasificador considerado denotado por $D_i \ \forall i = \{1, 2, \ldots, n\}$, que es igual a calcular cada $X_{C_{1_i}} - X_{C_{2_i}}$, podemos obtener la media de la diferencia de las muestras como:

$$M_D = \frac{\sum_{i=1}^{n} D_i}{n}$$

   Posteriormente se calcula desviación de la sumas cuadradas, SS_D, de la siguiente forma:

$$SS_D = \sum_{i=1}^{n} D_i^2 \times \frac{(\sum_{i=1}^{n} D_i)^2}{n}$$

2. En este paso estimamos la varianza de la población de las muestras como:

$$\{s^2\} = \frac{SS_D}{n-1}$$

3. Estimamos la desviación estándar de la distribución de las muestras M_D como:

$$\sigma_{M_D} = \sqrt{\frac{\{s^2\}}{n}}$$

   Esta fórmula desarrollada sería de la siguiente forma:

$$\sigma_{M_D} = \sqrt{\frac{SS_D/(N-1)}{n}}$$

4. Con estos elementos procedemos a calcular el valor de t de la siguiente forma:

$$t = \frac{M_D}{\sigma_{M_D}}$$

5. Obteniendo del paso anterior el valor de t, tomamos una nueva variable denotado por $df = n - 1$ y revisamos el nivel de significancia del conjunto de muestras tal y como se muestra en la Tabla 3.3:

Tabla 3.3: Valores críticos de t.

	Nivel de Significancia para una prueba direccional				
df	0.05	0.025	0.01	0.005	0.0005
1	6.31	12.71	31.82	63.66	636.58
2	2.92	4.30	6.96	9.92	31.60
3	2.35	3.18	4.54	5.84	12.92
4	2.13	2.78	3.75	4.60	8.61
5	2.02	2.57	3.36	4.03	6.87
6	1.94	2.45	3.14	3.71	5.96
7	1.89	2.36	3.00	3.50	5.41
8	1.86	2.31	2.90	3.36	5.04
9	1.83	2.26	2.82	3.25	4.78
10	1.81	2.23	2.76	3.17	4.59
11	1.80	2.20	2.72	3.11	4.44
12	1.78	2.18	2.68	3.05	4.32
13	1.77	2.16	2.65	3.01	4.22
14	1.876	2.14	2.62	2.98	4.14
15	1.75	2.13	2.60	2.95	4.07
16	1.75	2.12	2.58	2.92	4.01
17	1.74	2.11	2.57	2.90	3.97
18	1.73	2.10	2.55	2.88	3.92
19	1.73	2.09	2.54	2.86	3.88
20	1.72	2.09	2.53	2.85	3.85

Teniendo los valores de t y de df se puede ubicar el nivel de significancia del conjunto de muestra, en la Tabla 3.4 podemos ver que tenemos hasta el valor df = 20, esto es que podemos revisar el nivel de significacia hasta para 21 muestras, si se desea obtener este para un número de muestras mayor se puede encontrar en [19].

Para revisar como se calcula este método plantearemos el siguiente ejemplo con 15 muestras de dos clasificadores, C_1 y C_2, en donde cada ejemplo muestra las diferentes medidas de exactitud obtenidas por cada uno de ellos, los datos de estos se pueden ver en la Tabla 3.4:

Tabla 3.4: Ejemplo del cálculo de t-Test.

Muestra	$C:1$	C_2
1	71.2	74.2
2	70.4	69.4
3	72.8	71.6
4	75.2	77.0
5	71.8	75.5
6	75.7	76.6
7	70.2	72.0
8	71.5	71.2
9	71.1	73.4
10	76.9	71.1
11	74.6	73.8
12	70.4	70.2
13	70.8	74.1
14	77.1	77.6
15	71.0	75.8

Procedemos ahora a realizar los pasos para obtener el nivel de significancia de las muestras presentadas en la Tabla 3.5, calculando las diferencias de cada una de ellas, las cuales se muestran en la Tabla 3.5:

Tabla 3.5: Diferencias de C_1 - C_2.

Muestra	C_1	C_2	D_i
1	71.2	74.2	-3.0
2	70.4	69.4	+ 1.0
3	72.8	71.6	+ 1.2
4	75.2	77.0	- 1.8
5	71.8	75.5	- 3.7
6	75.7	76.6	- 0.9
7	70.2	72.0	- 1.8
8	71.5	71.2	+ 0.3
9	71.1	73.4	- 2.3
10	76.9	71.1	-4.2
11	74.6	73.8	+ 0.8
12	70.4	70.2	+ 0.2
13	70.8	74.1	- 3.3
14	77.1	77.6	- 0.6
15	71.0	75.8	- 4.8

Ahora procedemos a calcular la media de las diferencia de cada muestra y la desviación de la sumas cuadradas como sigue:

$$M_D = \frac{\sum_{i=1}^{n} D_i}{n} = -1.53$$

$$SS_D = \sum_{i=1}^{n} D_i^2 \times \frac{(\sum_{i=1}^{n} D_i)^2}{n} = 55.45$$

Una vez que tenemos estos valores podemos calcular la varianza y la desviación estándar:

$$\{s^2\} = \frac{SS_D}{n-1} = 3.96$$

$$\sigma_{M_D} = \sqrt{\frac{\{s^2\}}{n}} = \pm 0.51$$

Ahora que hemos calculado estos valores procedemos a calcular el valor de t de la siguiente forma:

$$t = \frac{M_D}{\sigma_{M_D}} = \frac{-1.53}{\pm 0.51} = \pm 3.0$$

Como tenemos 15 muestras, $n = 15$ por lo que $df = n - 1 = 14$, ahora solamente necesitamos revisar la Tabla 3.4, para ver cual es el nivel de significancia de este ejemplo. Con el valor de $df = 14$, el nivel de significancia para esta prueba direccional más cercano es 0.01, ya que el valor de t teórico es de $\pm$ 2.98, lo que nos dice que nuestro nivel de significancia es del 99.99 %. Con este ejemplo cerramos este capítulo, por lo que en el Capítulo 4 mostraremos cual fue nuestra propuesta para realizar la fase Minería de Datos en donde realizaremos la tarea de clasificación de datos.

Capítulo 4

Algoritmo de Mezcla Genética de Expertos Ponderada

4.1. Introducción

La Clasificación es una de las tareas más importantes en Minería de Datos, en esta tarea se busca utilizar la mejor forma de como poder distinguir las clases de cada uno de los ejemplos presentes en las BD.

Como pudimos observar en el Capítulo 2, existen diferentes algoritmos que implementan un tipo en específico de modelo de clasificación, de los cuales revisamos a detalle cinco modelos de clasificación los cuales fueron análisis de clusters (K - Medias), árboles de decisión (ID3, C4.5), clasificadores bsados en casos (k - Vecinos más Cercanos), clasificadores bayesianos (Naive Bayes) y reglas de decisión (Tablas de Decisión).

Estos forman una pieza importante en el desarrollo de nuestro modelo de clasificación, ya que nuestra propuesta consiste en implementar un Algoritmo de Mezcla Genética de Expertos Ponderada (*Weighted Genetic Mixture of Experts* - WGME), en el que se combinan varios tipo de clasificadores, aplicando un criterio de votación ponderada para obtener la clasificación del conjunto de prueba, esta propuesta cae dentro del esquema de los Clasificadores Basados en Acoplamientos.

El objetivo de estos modelos de clasificación, es construir un modelo a partir de la combinación de varios tipos de clasificadores, utilizando algún criterio de combinación para las clasificaciones individuales. Este tipo de clasificadores se divide en cuatro tipos de clasificadores, para este tra-

bajo escogimos un tipo de clasificadores llamados Mezcla de Expertos. Para poder entender el funcionamiento de los clasificadores basados en acoplamientos, la siguiente sección será dedicada a describir los cuatro tipos que existen.

4.2. Algoritmos de Clasificadores Basados en Acoplamientos.

Diferentes métodos para los Algoritmos de Clasificación por votación, como Bagging[5], AdaBoost[13] y Bootstrap[11], han mostrado tener una mejor exactitud con respecto a los clasificadores tradicionales, para BD reales como las proporcionadas por el EMCL PKDD. Los algoritmos de clasificación basados en acoplamientos se clasifican en cuatro tipos, *Bagging*, *Boosting*, Generalización Apilada y Mezcla de Expertos, cada uno de estos considera un conjunto de clasificadores, de los cuales se combinan las clasificaciones individuales para obtener la clasificación final del modelo considerado. Los dos criterios de combinación de clasificaciones individuales son los siguientes:

a) Votación por mayoría: Varios clasificadores diferentes votan para decidir la clase de un ejemplo en los datos.

b) Votación ponderada: Los clasificadores tienen distintos pesos en la votación (en función de su exactitud) y mediante este sistema se encuentra la clase de cada ejemplo.

A continuación describiremos más a detalle los cuatro tipos de clasificadores basados en acoplamientos.

4.2.1. *Bagging*

El termino *Bagging* viene del acrónimo "**b**ootstrap **agg**regat**ing**", en este tipo de modelos de clasificación, se consideran un grupo de clasificadores, por ejemplo varios árboles de decisión con diferentes estructuras, construidos a partir de diferentes conjuntos de prueba en la fase de entrenamiento. Una vez que la fase de entrenamiento concluye cada uno de los árboles presentes, predicen a que clase pertenece cada ejemplo de prueba. Es aquí donde un criterio de votación es aplicado,

contabilizando los votos para cada una de las clases presentes, escogiendo la clase que recibe más votos, esto es, la clase que fue predicha más veces por los árboles de decisión (Bootstrap[11]).

En el algoritmo de votación por mayoría básico es generado por un muestreo uniforme de casos m, tomados del conjunto de entrenamiento. Se generan T muestreos con remplazo (*bootstrap samples*), B_1, B_2, ..., B_T y un clasificador C_i es utilizado para cada uno de estos muestreos con remplazo B_i. Al finalizar cada una de las clasificaciones con los clasificadores C_1, C_2,..., C_T se construye un clasificador final C^* el cual contiene la clasificación del conjunto de prueba. A continuación se muestra el algoritmo básico para los métodos de votación por mayoría.

proc *Algoritmo de Votación por Mayoría* ($(S, I, T) \equiv$

(S: Conjunto de Entrenamiento, I: Clasificadores, T: # muestras con remplazamiento)

para $i = 1$ hasta T **haz**

$S' = muestrasconremplazamientodeS$

$C_i = I(S')$(Clasificación de cada clasificador)

termina

$C^*(*) = argmax_{y \in Y} \sum_{i:C_i(x)=y} 1$(Se construye la clasificación final)

.

4.2.2. *Boosting*

Para este modelo de clasificación se cuenta con varios modelos de un solo tipo de clasificador, generalmente se utilizan árboles de decisión, en donde cada uno de ellos tiene un peso asociado en base a la efectividad que presentan. El proceso es similar que el utilizado en votación por mayoría, pero en este caso cada voto tiene un peso mayor o menor según sea la ponderación que se le da a cada clasificador.

Se puede imaginar una elección presidencial mexicana, cada votante elige un candidato para darle su voto, cuando la elección termina, el ganador es el candidato con mayor número de votos. El tamaño de los estados, conducen el rumbo de las elecciones, ya que un estado de gran tamaño implica un mayor número de votantes, por ello los candidatos realizan una mayor campaña en esos estados ya que aseguran un buen porcentaje de los votos totales. Tomando este ejemplo pode-

mos hacer una similitud con los modelos de clasificación por votación ponderada, los diferentes clasificadores representan a los votantes, en donde cada uno de ellos tienen un peso en relación a su efectividad, finalmente mediante una votación final se asigna la clase a la que pertenece cada ejemplo de prueba (candidato de la elección).

Para ver el funcionamiento de este tipo de algoritmos por votación ponderada, tomaremos el algoritmo AdaBoost[13] el cual genera una clasificación mediante la votación de un conjunto de clasificadores. En los 90's Shapire demostró que si se utilizan aprendices débiles, de los cuales se obtienen una serie de clasificaciones individuales, si se encontraba una forma de como combinar estas clasificaciones individuales, estos aprendices pueden ser transformados en aprendices fuertes. En aquellos momentos estos resultados fueron considerados como un algoritmo seminal para Aprendizaje Automático[27].

La construcción de este tipo de algoritmos es similar a Bagging, la única diferencia es que solo se considera árboles de decisión para su construcción. La operación de este es algoritmo es la siguiente, se realiza la ejecución secuencial de los diferentes clasificadores, algo diferente a los algoritmos de votación por mayoría que lo realizan de manera paralela. El algoritmo AdaBoost selecciona diferentes tamaño de los conjuntos de entrenamiento, los cuales nos van a servir para tener un entrenamiento de cada uno de los clasificadores antes de que se construya la clasificación final. La meta es forzar al clasificador para que reduzca el error mínimo de la predicción sobre diversas distribuciones de la entrada del conjunto de entrenamiento.

Dado un número entero T que especifica el número de pruebas, se generan en orden T conjuntos de entrenamiento denotados por S_1, S_2, ..., S_T y ademas se construyen los T clasificadores denotados por C_1, C_2, ..., C_T . Al final se construye un clasificador C^* tomando cada uno de los clasificadores $C_i \; \forall \; i = \{1, 2, \ldots, T\}$, aplicando un esquema de la votación ponderada. El peso de cada clasificador depende de su exactitud obtenida para los diferentes conjuntos de entrenamiento en la fase de entrenamiento. El algoritmo de AdaBoost es mostrado a continuación:

proc *AdaBoost* ($(S, I, T) \equiv$

(S: Conjunto de Entrenamiento de tamaño m, I: Clasificadores, T: # de pruebas)

$S' = S$

para $i = 1$ hasta T **haz**

$C_i = I(S')$(Clasificación de cada clasificador)

$\epsilon_i = \frac{1}{m} \sum_{x_j \in S': C_i(x_j) \neq j_i} peso(x)$ (error ponderado en el conjunto de entrenamiento).

si $\epsilon_i > 1/2$ **entonces**

(Fijamos S' a una muestra con remplazamiento de S con un peso de 1 para cada caso y regresamos al inicio del ciclo **para**, usualmente esto se fija el numero T=25 .)

termina

$\beta_i = \epsilon_i/(1 - \epsilon_i)$

para cada $x_j \in S'$

si $C_i(x_j) = y_j$ **entonces** $peso(x_j) = peso(x_j) \times \beta_i$

Se normalizan los pesos de los casos para que el peso total de S' sea m.

termina

termina

$C^*(*) = argmax_{y \in Y} \sum_{i: C_i(x) = y} log \frac{1}{\beta_i}$

.

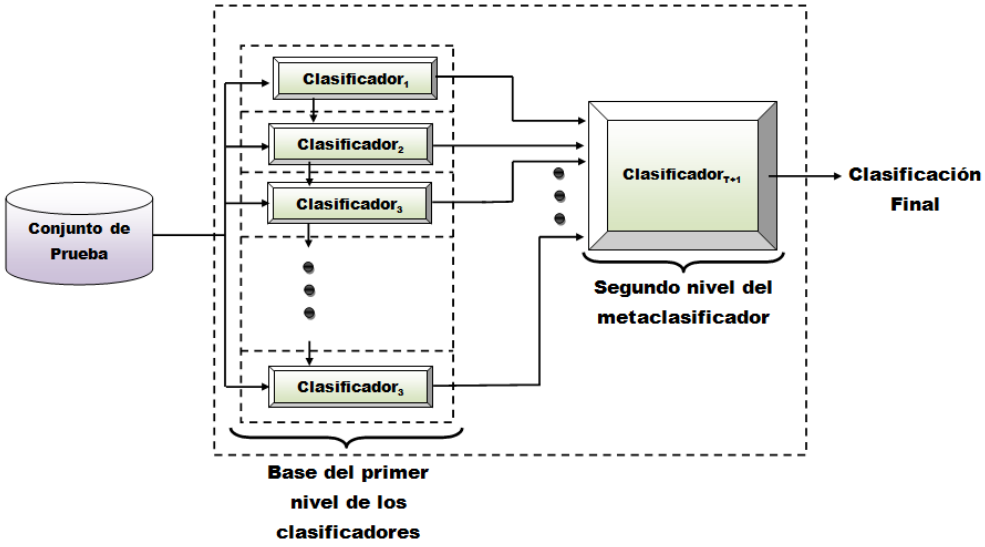
4.2.3. Generalización Apilada *Stacked Generalization*:

Este método fue introducido por Wolpert[37], usando un conjunto de clasificadores denotados por C_1, C_2, C_3, ..., C_T, los cuales son entrenados previamente por diferentes conjuntos de entrenamiento, de los cuales se obtienen una serie de clasificaciones individuales para cada uno de los ejemplos en el conjunto de prueba. Esta primera fase de este tipo de clasificadores es llamado la Base del primer nivel de los clasificadores.

Después de obtener las clasificaciones individuales, se selecciona algún criterio de combinación

de clasificaciones individuales, para este tipo de clasificadores se escoge el de votación por mayoría, en el cual cada uno de los clasificadores presentes emiten un voto hacia algunas de las clases presentes en los datos. Al finalizar el primer nivel de los clasificadores cada una de las salidas de los clasificadores seleccionados, corresponden a las entradas de un nuevo clasificador llamado metaclasificador, en el que se realiza un conteo de cada uno de los votos y se escoge la clases con mayor número de votos para cada uno de los ejemplos en los datos, a esta fase se conoce como el segundo nivel del metaclasificador, la cual es la parte final de este tipo de clasificadores basados en acoplamientos. El esquema de funcionamiento de este tipo de clasificadores se puede ver en la Figura 4.1.

Figura 4.1: Esquema de funcionamiento de la generalización apilada.



4.2.4. Mezcla de Expertos *Mixture of Experts:*

Este tipo de clasificadores basados en acoplamientos es similar a los clasificadores de la generalización apilada, considerando un grupo de clasificadores denotados por C_1, C_2, C_3, ..., C_T, para realizar la bases del primer nivel de estos clasificadores, para obtener el clasificador C_{T+1}, en el que se combinan las clasificaciones individuales de cada uno de los clasificadores considerados.

En este tipo de clasificadores, a diferencia de la generalización apilada, se utiliza un método para encontrar la ponderación de cada unos de los clasificadores usados, esto porque se implementa un criterio de combinación de clasificaciones individuales del tipo votación ponderada. Usualmente se utilizan Redes Neuronales para entrenar a este tipo de clasificador y obtener la mejor ponderación para cada clasificador, esta red neuronal es llamada Gating Network[27].

Una vez que se tienen las ponderaciones de cada uno de los clasificadores emiten un voto por cada ejemplo en los datos, ahora se contabilizan los votos, los votos de cada clasificador pueden ser diferentes o iguales unos de otros ya que ahora se tiene un ponderación que da un peso diferente a cada uno de los clasificadores de acuerdo a su grado de exactitud, es por ello que este tipo de clasificadores reciba el nombre de Mezcla de Expertos, con el termino de esta votación se obtienen las clasificaciones finales del conjunto de datos considerados.

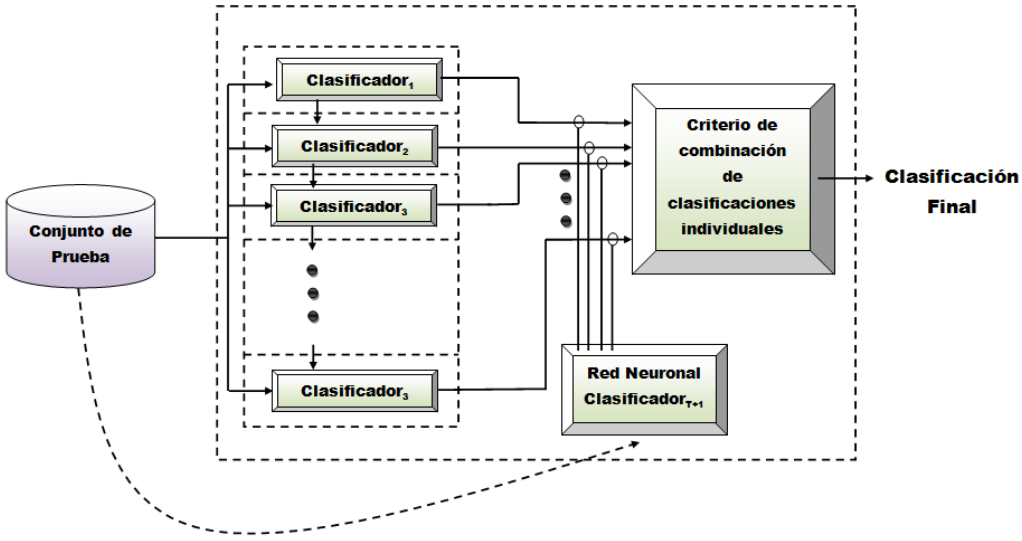
Este tipo de clasificadores es importante para el trabajo que presentamos, ya que nuestra propuesta consiste en desarrollar un algoritmo basado en acoplamientos de este tipo, en el que consideramos una serie de los mejores clasificadores que encontramos al realizar una serie de pruebas. Implementamos un algoritmo genético para encontrar la mejor ponderación para cada clasificador, finalmente consideramos un criterio de votación ponderada para encontrar la clasificación final de los datos, este algoritmo sera descrito en la siguiente sección. El esquema de funcionamiento de la mezcla de expertos se puede ver en la Figura 4.2.

4.3. Construcción del WGME.

El algoritmo que se propuso, es novedoso ya que considera un grupo de varios tipos de clasificadores y no solo a un tipo en especial, por ejemplo árboles de decisión[1, 32, 21, 11, 5, 13], para construir su modelo de clasificación y el hecho de usar un algoritmo genético para encontrar la mejor ponderación de cada clasificador considerado, obteniendo una mejora en la exactitud, mediante la combinación de las clasificaciones individuales de cada uno de ellos.

Para poder realizar nuestra propuesta, el primer paso que se llevo a cabo fue la prueba individual de un conjunto de clasificadores, para llevar a cabo el problema de la longitud de la vista, revisado en el Capítulo 3. Este es un problema de clasificación de datos, tomando diferentes tamaños del

Figura 4.2: Esquema de funcionamiento de la mezcla de expertos.



conjunto de prueba, de la BD gemius_completa. En la parte final del Capítulo 3 quedo pendiente la fase de Minería de Datos del proceso de KDD, en esta sección de este capítulo describiremos cual fue la propuesta desarrollada para llevar acabo este problema de clasificación.

De acuerdo a J. Ross Quinlan[38] et al. en el articulo 'Top 10 algorithms in data mining', los clasificadores de mayor influencia en la comunidad de investigadores de Minería de Datos, son los presentados en la Tabla 4.1.

Dada su relevancia, en un principio fueron considerados para realizar las pruebas, con las diferentes herramientas, WEKA, SIPINA y TANAGRA. Aunque nos presentamos con el problema de que estas herramientas para Minería de Datos no consideraban algunos de ellos como PageRank, Apriori y EM, además en algunos de ellos, se presentaba un tiempo de ejecución muy grande para el tamaño del conjunto de pruebas que utilizamos como por ejemplo SVM.

Se consideraron este conjunto de clasificadores y además se seleccionaron otros más, buscando aquellos que tuvieran presentaran un mejor rendimiento, en especifico con la medida de exactitud. Finalmente seleccionamos un conjunto más reducido, el cual se puede ver en la Tabla 4.2.

Una vez que se seleccionaron a los clasificadores que estarían presentes en nuestro WGME,

Tabla 4.1: Clasificadores más utilizados.

Case	*Name*	*Type of learning*
1	*K*-Medias	Aprendizaje no supervisado
2	*k*-Vecinos más Cercanos	Aprendizaje Supervisado
3	Bayes Ingenuo	Aprendizaje Supervisado
4	AdaBoost	Aprendizaje Supervisado
5	SVM (*Support Vector Machine*)	Aprendizaje Supervisado
6	Apriori	Aprendizaje Supervisado
7	C4.5	Aprendizaje Supervisado
8	CART	Aprendizaje Supervisado
9	PageRank	Aprendizaje Supervisado
10	EM(Expectation−Maximization)	Aprendizaje Supervisado

Tabla 4.2: Lista de clasificadores seleccionados.

Case	*Name*	*Type of learning*
1	Bayes Ingenuo	Aprendizaje supervisado
2	*k*-Vecinos más Cercanos	Aprendizaje Supervisado
3	Decision Tables	Aprendizaje Supervisado
4	ADTree	Aprendizaje Supervisado
5	C4.5	Aprendizaje Supervisado
6	*K*-Medias	Aprendizaje Supervisado

se tenían que implementar cada uno de los clasificadores vistos en la Tabla 4.2 y otros más como, SVM y Perceptrón Multicapa, para obtener la clasificación individual de cada uno de ellos. Ademas se tenía que implementar un algoritmo que implementara nuestra propuesta en el que se considerará la clasificación en conjunto de cada uno de los clasificadores vistos en la Tabla 4.2. Esto se realizó sobre Matlab 7.7 (2008b), siguiendo los siguientes tres pasos:

1. Primero se aplicaba alguno de los clasificadores de la Tabla 4.2 a un conjunto de entrenamiento, de ahí obteníamos una clasificación que guardábamos en un archivo para hacer uso de ella posteriormente. Esto lo hacíamos con cada uno de los clasificadores considerados.

2. Después de clasificar de manera individual, recolectábamos cada clasificación individual para formar una matriz de tamaño $n \times m$, donde n era el número de ejemplos en el conjunto de prueba y m el número de clasificadores seleccionados, la cual llamaremos *matriz de ponderaciones*.

3. Ya que se tenía esta matriz se procedía a tomar una serie de pesos arbitrianos para encontrar la clasificación final de cada ejemplo mediante un criterio de votación ponderada.

Podemos observar en este método de clasificación los siguientes dos inconvenientes:

- Los pesos seleccionados se realizaban al azar, lo cual no nos garantizaba que fuera la mejor manera de hacerlo.

- Todos las implementaciones trabajaban por separado y no existía una forma en la cual todo estuviera en un solo método, que llevara el control de las clasificaciones de los diferentes métodos y la clasificación final del conjunto de prueba mediante el criterio de votación ponderada.

En las Secciones 4.3 y 4.4 explicaremos como resolvimos estos dos inconvenientes, para tener una versión final de la aplicación que sera descrita a detalle en la Sección 4.5.

4.4. Criterio de selección de Pesos

El criterio de selección de pesos en un principio era asignar de una forma arbitraria un peso a cada clasificador, pero esto no resultaba ser la forma de hacerlo. Con diferentes pesos, se obtenían diferentes valores de exactitud, ¿Pero cómo saber cuál es la mejor configuración?, la respuesta que encontramos a esta pregunta fue la de aplicar un simple algoritmo genético.

Los algoritmos genéticos fueron inventados por John Holland[17] por la década de los 70, inspirados por el proceso de la evolución. En este tipo de algoritmos se parte de una población inicial, la cual se va recombinando mediante las operaciones como mutación, selección y cruzamiento para así dar una solución a algún problema de optimización dado. Se puede considerar a este tipo

de algoritmos como de búsqueda dispersa ya que se busca un máximo o mínimo, según sea el problema, sobre un espacio de búsqueda descrito por el mismo. El pseudocódigo de un algoritmo genético es el siguiente:

proc *Algoritmo_Genético*

$t \leftarrow 0, t =$ iteración

incializa $P(t), P =$ población

evalúa $P(t)$

mientras no se cumpla condición de paro

$t \leftarrow t + 1$

selecciona padres de $P(t-1)$

cruzamiento $P(t-1)$ con una probabilidad de cruza P_c

mutación $P(t-1)$ con una probabilidad de cruza P_m

evalúa $P(t)$

termina

.

Las características del algoritmo genético que implementamos para la selección de los pesos de cada cromosoma se muestra a continuación:

- En el algoritmo que implementamos, cada población representa los pesos de cada clasificador de la Tabla 4.2, como se escogieron seis clasificadores distintos, el tamaño de cada cromosomas es de seis.

- La codificación de cada cromosoma, tiene un peso en un rango de [0, 0.5, 1, 1.5, 2, . . . , 4], un ejemplo de una población para este algoritmo genético se puede ver en la Tabla 4.3.

- La función de aptitud que se escogió para la evaluación de cada cromosoma, es el valor de exactitud que tiene cada cromosoma, por lo que nuestro objetivo es resolver un problema de maximización del valor de exactitud. Para las nuevas generaciones se seleccionan aquellos que representan una mejor exactitud.

- Se aplica el operador de cruzamiento en un solo punto, para poder llevar a cabo la recombi-

Tabla 4.3: Codificación de los cromosomas.

Cromosoma	*Bayes Ingenuo*	*k-Vecinos más Cercanos*	*Decision Table*	*ADTree*	*C4.5*	*k-Medias*
1	3	2	0.5	2	2.5	4
2	2	3.5	1.5	3	1.5	0.5
3	3.5	2.5	1	4	2	2
4	4	0.5	4	1	1.5	1
5	1.5	1.5	3	1.5	3.5	0.5
…	…	…	…	…	…	…
n	0.5	2.5	1	4	0.5	1.5

nación genética de cada una de las poblaciones que se generen.

- Como se esta utilizando una BD *larga* con un total de 379,485 ejemplos, se selecciona de forma aleatoria un 10 % del total, para tener un conjunto de prueba y de esta forma poder realizar las pruebas con este algoritmo genético.

- Se aplica el criterio de elitismo, para que el algoritmo converge en probabilidad al óptimo global.

- El criterio de paro utilizado para este algoritmo genético, es el del número de iteraciones. Se corrió 500 veces este algoritmo, hasta que se encontró la mejor forma de asignarle los pesos a cada uno de los clasificadores presentes, la cual se muestra en la Tabla 4.4.

De esta forma fue como resolvimos la forma de asignar un peso a cada uno de los clasificadores, encontrando una forma novedosa de hacerlo al aplicar este algoritmo genético.

4.5. Funcionamiento del WGME

La segunda inconveniente que teníamos, era que cada uno de los clasificadores se encontraba implementado por separado, así como el motor de trabajo de nuestra propuesta, no se encontraba trabajando en una forma correcta. Es por esto que decidimos juntar cada uno de los clasificadores de la Tabla 4.2, en un método que llevara el control de cada uno de estos clasificadores, de esta

Tabla 4.4: Pesos asignados a los clasificadores considerados.

Caso	*Nombre*	*Ponderación*
1	Bayes Ingenuo	4
2	*k*-Vecinos más cercanos	0.5
3	Tablas de Decisión	1
4	ADTree	2
5	C4.5	1
6	*K*-Medias	0.5

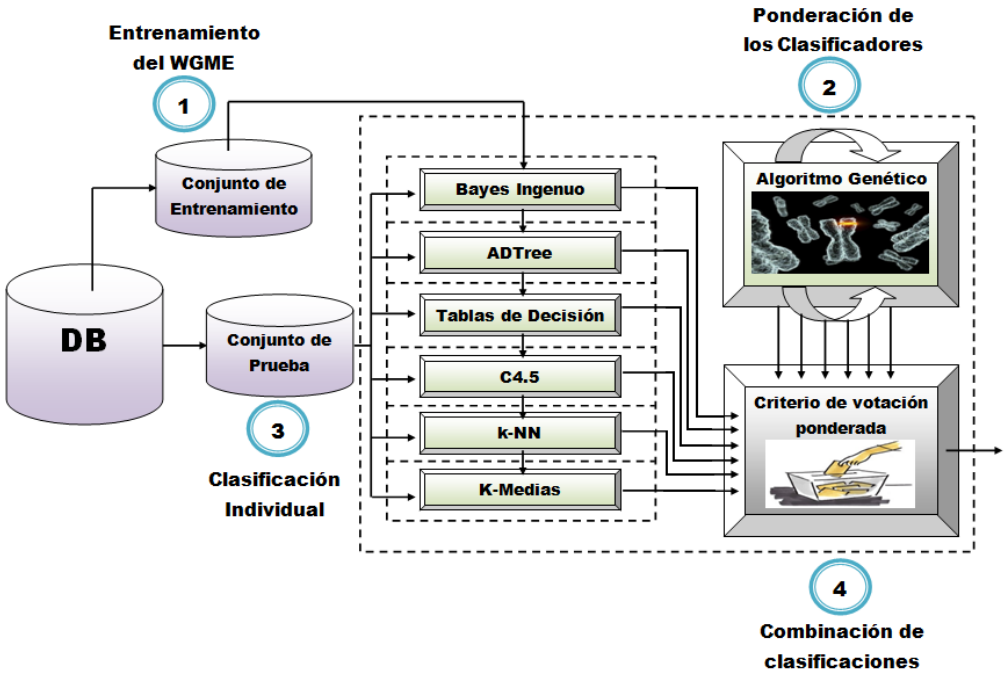
manera se puede realizar la clasificación de cualquier tamaño del conjunto de pruebas. El esquema de funcionamiento de este modelo se puede ver en la Figura 4.3.

A continuación describiremos los tres pasos para llevar acabo la clasificación de los datos con nuestra propuesta:

1. **Entrenamiento del WGME:** De la BD seleccionamos un conjunto de prueba, que nos va a servir para entrenar a cada uno de los clasificadores presentes en nuestro modelo de clasificación propuesto. Dependiendo del tamaño que escojamos para nuestro conjunto de prueba, los clasificadores recibirán un conjunto de entrenamiento del mismo tamaño o mayor que el conjunto de prueba. Como sabemos el clasificador K-Medias, es un clasificador de aprendizaje no supervisado, para este caso se utiliza solo el conjunto de prueba, del cual se desconoce la clase de cada uno de los ejemplos. El objetivo en este caso es construir n clusters dependiendo del número de clases que se encuentren presentes en los datos, para la BD gemius_completa, n = 2, ya que tenemos solo dos clases, por lo que se construyen dos clusters para separar a cada uno de los ejemplos y de esta manera conseguir la clasificación de los datos mediante este clasificador.

2. **Ponderación de los Clasificadores:** En este punto se genera un conjunto de prueba cuyo tamaño es del 10 % (este porcentaje es solo para esta BD, pero puede ser modificado depen-

Figura 4.3: Esquema de funcionamiento del WGME.



diendo de la BD que se utilice) del tamaño total de BD gemius_completa, para poder encontrar la mejor ponderación para cada uno de los clasificadores utilizados. Como describimos en la Sección 4.4, al ejecutar 500 veces este algoritmo encontramos que la mejor configuración de pesos fue la que se mostró en la Tabla 4.4, por lo que se puede o no realizar esta parte del algoritmo, ya sea generando una configuracion distinta o utilizando directamente la configuración de la Tabla 4.4.

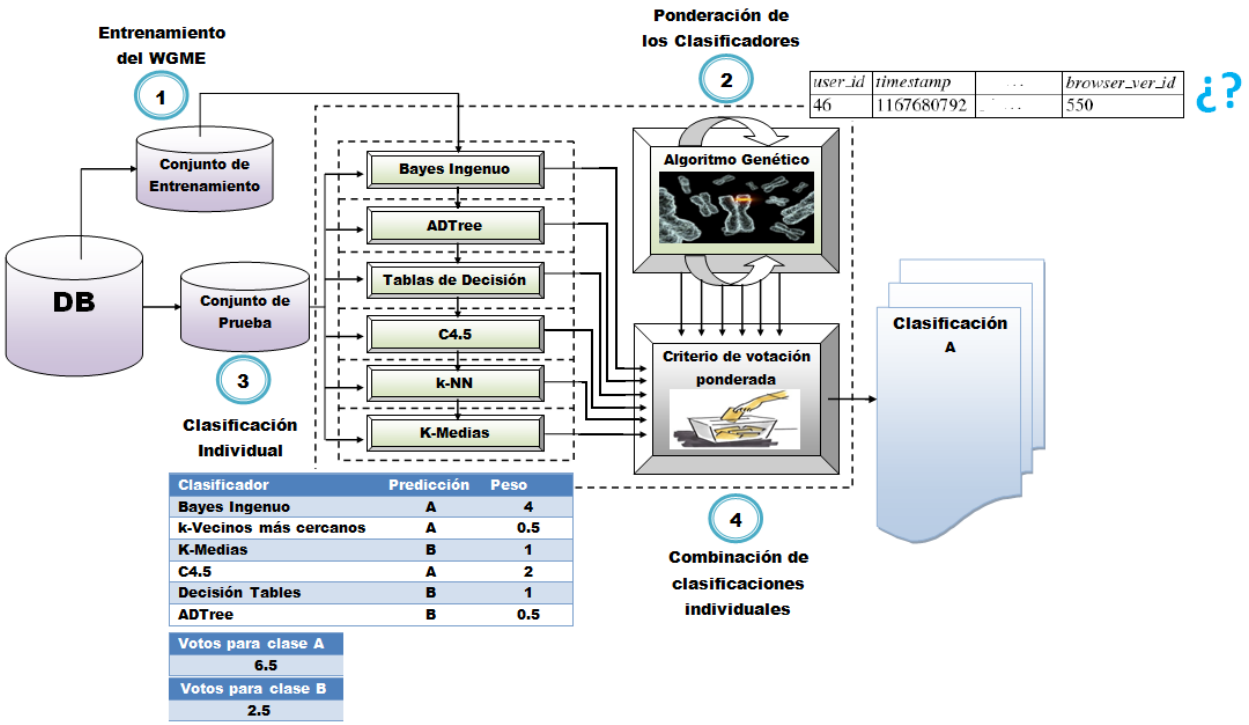
3. **Clasificación Individual:** En esta parte, se realiza la clasificación individual del conjunto de prueba, con cada uno de los clasificadores. Al terminar cada uno su clasificación se va almacenando de manera automática, hasta formar la matriz de ponderaciones, en la cual se encentraran las clasificaciones individuales de cada uno de los clasificadores.

4. **Criterio de Votación Ponderada:** El momento en el que se forma la matriz de ponderaciones, se puede aplicar el criterio de votación ponderada, para encontrar la clasificación final del conjunto de prueba. Una vez que tenemos esta clasificación se pueden calcular las

diferentes medidas de rendimiento para Minería de Datos, que discutiremos en el Capítulo 5.

Un ejemplo de como se lleva a cabo esta clasificación se puede ver en el esquema de la Figura 4.4.

Figura 4.4: Ejemplo de funcionamiento del WGME.



En este caso podemos ver en el esquema de la Figura 4.4, que se elige a un ejemplo de prueba, del cual desconocemos la clase a la que pertenece, revisamos las predicciones individuales de cada uno de los clasificadores para este ejemplo de entrenamiento y aplicamos el criterio de votación, contabilizando los votos para la clase A y la clase B. En este caso la clase A tiene 6.5 votos y la clase B tiene 2.5, seleccionamos la de mayor número de votos y se la asignamos al ejemplo de prueba, en este caso asignamos la clase A. El esquema de la Figura 4.4, muestra un ejemplo de como se realiza la clasificación de un conjunto de prueba por medio del WGME propuesto, ahora en la siguiente sección veremos cual es la aplicación que desarrollamos en Matlab 7.7 (2008b).

4.6. Aplicación desarrollada

En esta aplicación se implementaron todos los clasificadores de la Tabla 4.2 y algunos otros más, en Matlab 7.7 (2008b), los cuales nos permiten realizar la meta de clasificación de datos. Cada una de estas implemetaciones están construidas como una función, en donde los parámetros de entrada son los conjuntos de entrenamiento y de prueba. Un ejemplo de como se definen estos clasificadores es el siguiente:

```
function test_targets = Bayes_Ingenuo(handles,train_patterns,
         train_targets,test_patterns, real_test_targets)
```

Donde:

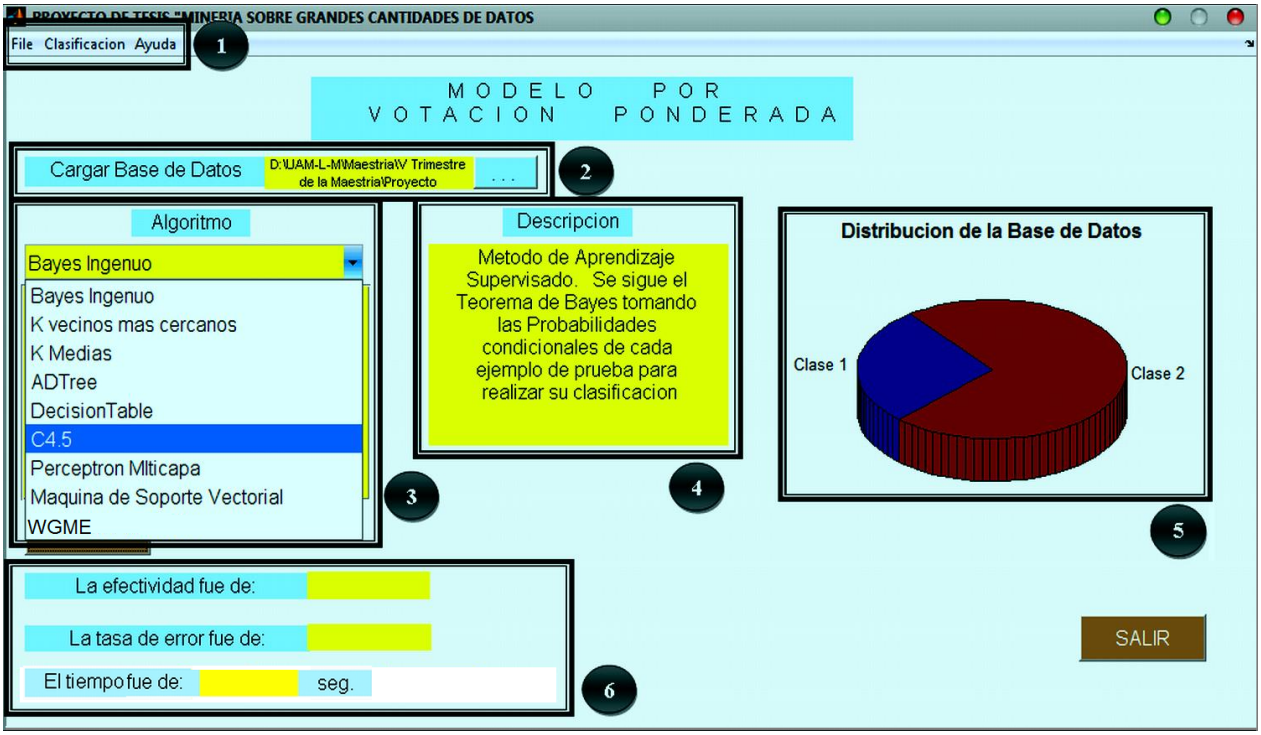
- *function*: Es la forma de indicar en Matlab que se trata de un función.
- $test\_targets$: Es el dato que será regresado por la función.
- $Bayes\_Ingenuo$: El nombre que le asignamos a la función, en este caso esta función es para el clasificador Bayes Ingenuo.
- *handles*: Esto indica que se esta recibiendo como parámetro de entrada, un objeto, este tipo de variables nos permiten el manejo de interfaces gráficas, mediante la herramienta *GUIDE*.
- $train\_patterns$ y $train\_targets$: $train\_patterns$ contiene el conjunto de entrenamiento sin la clase a la que pertenece cada uno de ellos, $train\_targets$ contiene las clases de cada uno de los ejemplos de entrenamiento.
- $test\_patterns$ y $real\_test\_targets$: $test\_patterns$ contiene el conjunto de prueba sin la clase a la que pertenece cada uno de ellos, $real\_test\_targets$ contiene las clases de cada uno de los ejemplos de prueba para calcular las medidas de rendimiento con las clases encontradas por este clasificador, cuando se seleccione la clasificación solo con este clasificador.

Para cada uno de los clasificadores se tiene un formato similar, por lo que cada implementación representa un función dentro de la aplicación desarrollada. Para la función que implementaba el WGME, tenemos lo siguiente:

```
function test_targets = Votacion_Ponderada(handles,train_patterns,
         train_targets,test_patterns, real_test_targets)

%Llamado de cada clasificador
  matriz_ponderaciones(:,1) = Bayes_Ingenuo(handles,train_patterns,
  train_targets,test_patterns, real_test_targets)

  matriz_ponderaciones(:,2) = knn(handles,train_patterns,
  train_targets,test_patterns, real_test_targets,k)

  matriz_ponderaciones(:,3) = decision_table(handles,train_patterns,
  train_targets,test_patterns, real_test_targets)

  matriz_ponderaciones(:,4) = adtree(handles,train_patterns,
  train_targets,test_patterns, real_test_targets)

  matriz_ponderaciones(:,5) = C4_5(handles,train_patterns,
  train_targets,test_patterns, real_test_targets)

  matriz_ponderaciones(:,6) = K_medias(handles,test_patterns,
                              real_test_targets)

%Aplicacion del critero de votación ponderada
test_targets = motor_vp(handles,matriz_ponderaciones,
                          real_test_targets)
```

Podemos ver que se siguen los pasos presentados en la parte final de la Sección 4.4 y al final se obtiene la clasificación del conjunto de prueba con el WGME propuesto. El aspecto visual de esta aplicación se puede ver en la Figura 4.5:

Figura 4.5: Aspecto visual de la aplicación desarrollada.



Se puede observar que esta aplicación presenta una interfaz gráfica, desarrollada mediante la herramienta *GUIDE* de Matlab. *GUIDE* es un entorno de programación visual disponible en MATLAB para realizar y ejecutar programas que necesiten una interfaz visual. Tiene las características básicas de todos los programas visuales como Visual Basic o Visual C++. En la Figura 4.5 podemos ver que contamos con las siguientes características:

1. **Menú de Herramientas:** Esta ubicado en la parte superior derecha y nos da las opciones abrir una BD, visualización de la distribución de la BD, comenzar la clasificación de los datos, menús de ayuda, entre otros más.

2. **Cargar Base de Datos:** Nos permite seleccionar cual BD vamos a utilizar para empezar el proceso de clasificación.

3. **Algoritmo:** Es una lista de opciones en las que podemos elegir el clasificador que deseamos utilizá para llevar a cabo la clasificación de la BD seleccionada.

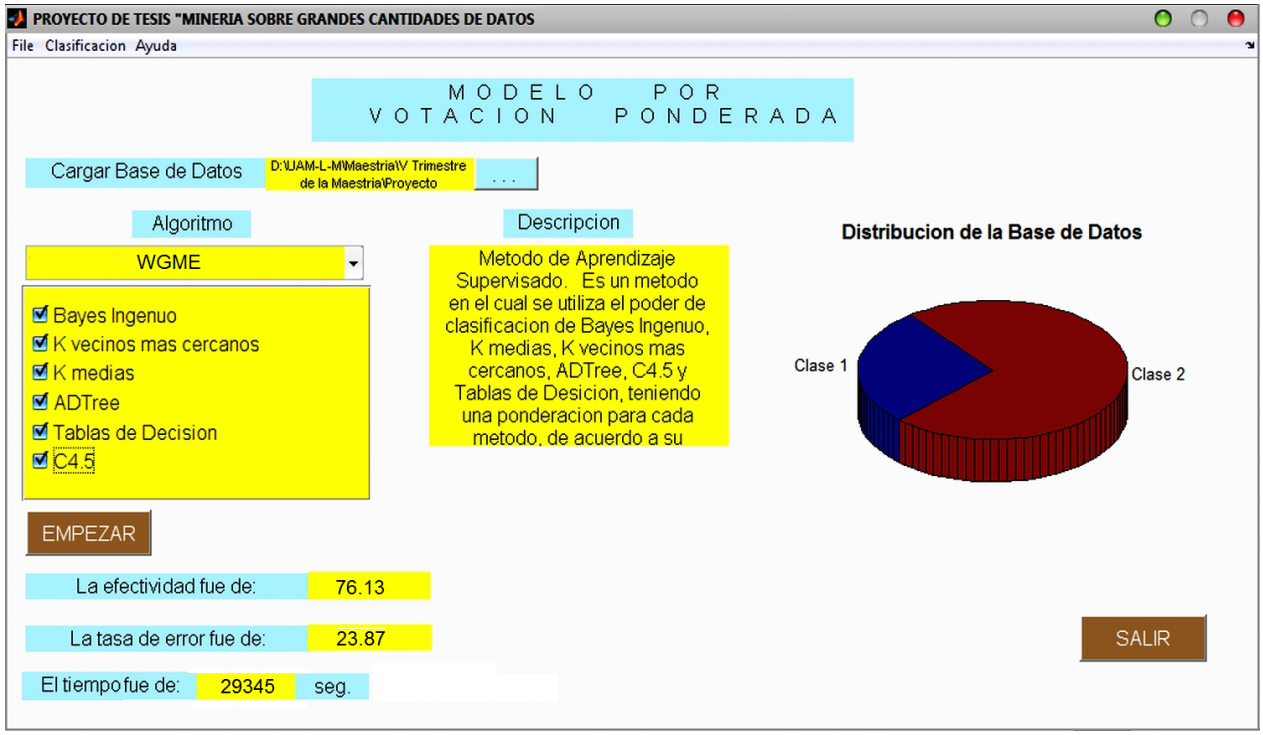
4. **Descripción:** Aquí se menciona el tipo de aprendizaje al que pertenece, así como una pequeña descripción del algoritmo.

5. **Distribución de la Base de Datos:** Se muestra una gráfica con el porcentaje que tiene cada una de las clases presentes en BD.

6. **Área de resultados:** Para cada clasificador seleccionado se muestra la efectividad que obtuvo y la tasa de error, también se muestra el número de hilos utilizados, mediante el uso de la programación multihilos, implementada en la versión de 7.7 de Matlab.

La opción del MVBA tiene cada uno de los clasificadores vistos en la Tabla 4.2, se pueden escoger todos para hacer la clasificación con el MVBA, o seleccionar con los que queremos que se realice la clasificación del conjunto de prueba. La Figura 4.6 muestra esta opción en donde se realizó una prueba utilizando los seis clasificadores:

Figura 4.6: Ejemplo de la aplicación desarrollada.



En este capítulo se mostró los pasos que seguimos para poder desarrollar el WGME propuesto, en el Capítulo 5, describiremos a detalle cuales fueron las pruebas que se realizaron con las diferentes herramientas consideradas, así como la comparación de estos resultados obtenidos, contra los obtenidos al aplicar las mismas pruebas pero con nuestra propuesta.

Capítulo 5

Pruebas y Resultados

5.1. Introducción

Las pruebas que presentaremos son las que se realizaron para dar solución al problema la longitud de la vista, el cual es un problema de clasificación, en donde debemos de ser capaces de asignar la clases a cada ejemplo del conjunto de prueba, que tomaremos de la BD gemius_completa, que se mostró en la Tabla 3.3 del Capítulo 3, Sección: Aplicación de la Fase de Selección. Como se comentó en el Capítulo 3, las herramientas que se seleccionaron para realizar las pruebas fueron, WEKA, SIPINA, TANAGRA e implementaciones propias de algunos clasificadores. A continuación damos una breve semblanza de estas herramientas.

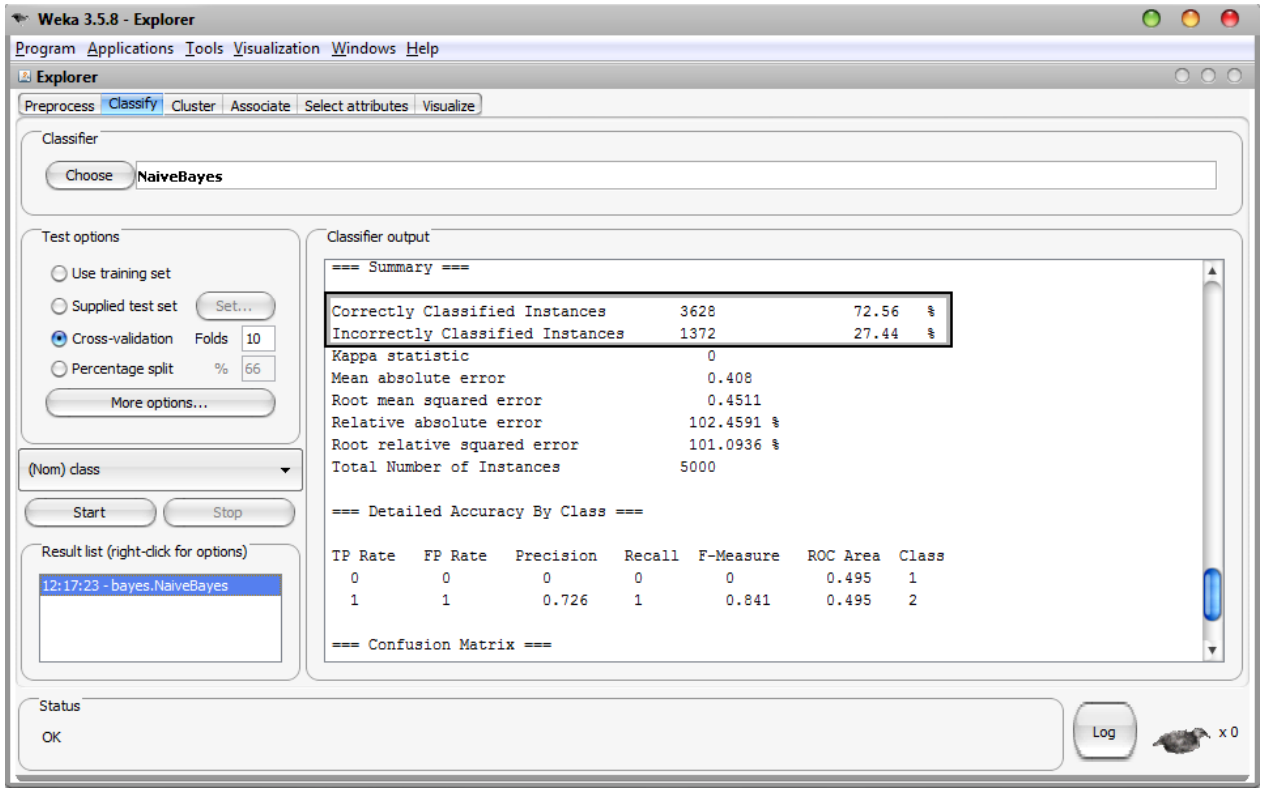
5.1.1. WEKA (Waikato Environment for Knowledga Analysis).

WEKA[36] fue desarollado por la Universidad de Waikato, situada en *Hamilton*, Nueva Zelanda. Es un Software libre para Minería de Datos, desarrollado en código Java, en el que se encuentran algunas implementaciones de varios clasificadores como: ID3, C4.5, CN2,ADTree, Decisión Table, Perceptrón Multicapa, Bayes Ingenuo y otros más, técnicas de visualización de datos, flujos de conocimiento, entre otros componentes que nos permiten llevar a cabo las tareas de Minería de Datos (predicción, identificación, agrupamiento, asociación y clasificación).

Esta herramienta fue considerada debido a que en muchos de los trabajos sobre Minería de Datos se le menciona como una de la principales herramientas, ya que nos permite realizar diversas pruebas que competen la tarea de llevar a cabo Minería de Datos, en la que usamos nosotros que es la de clasificación de datos. La facilidad de uso es otro de los factores por la que escogimos esta herramienta, ya que se nos proporciona una serie de documentación, con la que podemos empezar a trabajar, viendo como se configuran cada uno de los clasificadores que se encuentran implementados dentro de esta herramienta, para poder llevar a acabo la clasificación de los datos, tendremos que ir a la parte de exploración, tal y como se muestra en la pantalla como de la Figura 5.1.

En esta pantalla tendremos la opción de preprocesamiento para poder cargar nuestra BD con diferentes formatos como SQL y $ARFF$ que son los más utilizados, una vez que hayamos cargado nuestra BD prodremos pasar a la opción de clasificar en donde podemos llevar a cabo la clasificación con alguno de los clasificadores implementados. Los diferentes tipos de clasificadores con los que cuenta esta herramienta son clasificadores bayesianos, clasificadores por funciones, clasificadores difusos, clasificadores por reglas, clasificadores por árboles de decisión entre otros contando con un gran número de métodos para cada uno de ellos. En la Figura 5.1 presentaremos un ejemplo visual de como podemos realizar la clasificación de los datos mediante el método de Bayes Ingenuo, tomando la BD gemius_completa.

Figura 5.1: Ejemplo de clasificación con Bayes Ingenuo de WEKA

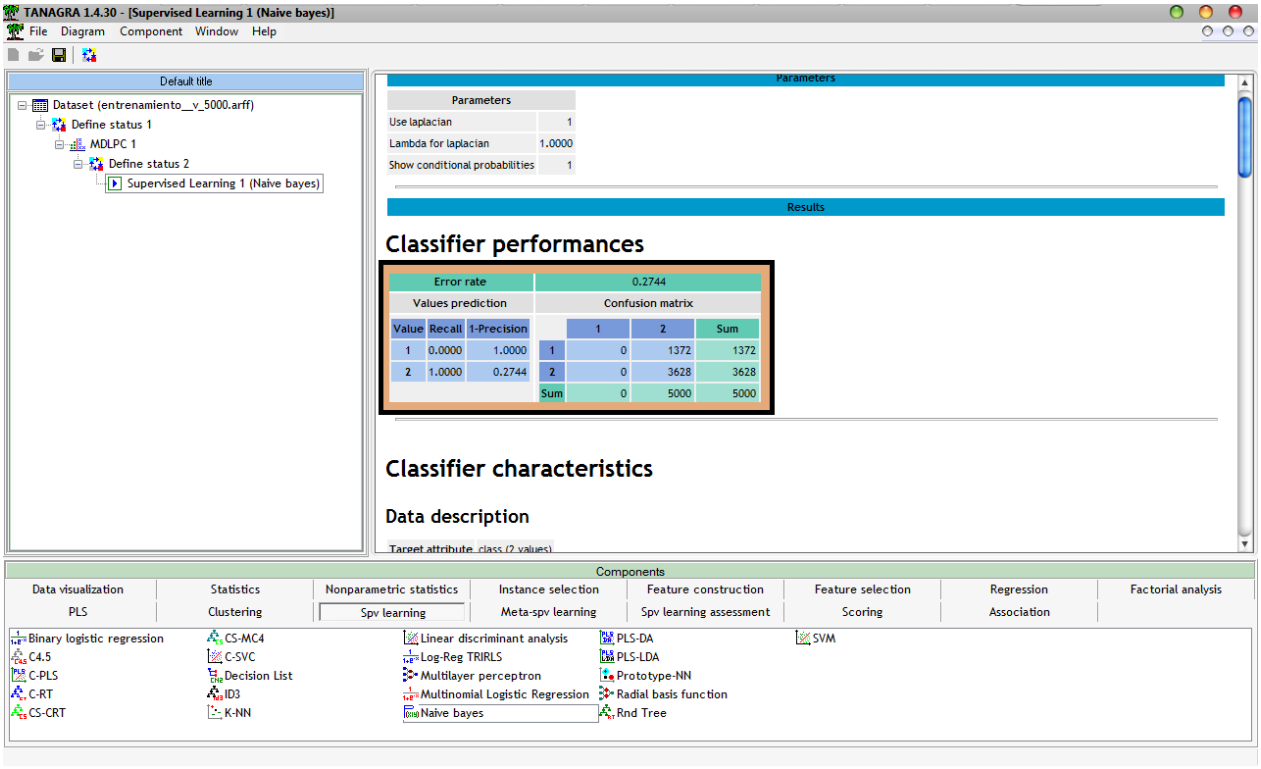


En la Figura 5.1 se puede observar, sobre la parte resaltada con un rectángulo, la clasificación que se obtuvo del conjunto de prueba seleccionado, mediante el método de Bayes Ingenuo.

5.1.2. TANAGRA (A free data mining software for research and education).

TANAGRA[36] fue desarrollado por Ricco Rakotomalala en Lyon, Francia. Es un Software libre para la aplicación de Minería de Datos, en esta herramienta se encuentra algunas implementación de clasificadores como *k*-Vecinos más Cercanos, Perceptrón Multicapa, ID3, Bayes Ingenuo, *K*-Medias, por mencionar solo algunos. En esta herramienta se pueden llevar a cabo las diferentes tareas de Minería de Datos, mostraremos un ejemplo de como se lleva a cabo la clasificación con el método de Bayes Ingenuo, aplicado a un conjunto de prueba tomado de la BD gemuis_completa, esto se puede observar en la Figura 5.2:

Figura 5.2: Ejemplo de la clasificación con Bayes Ingenuo de TANAGRA



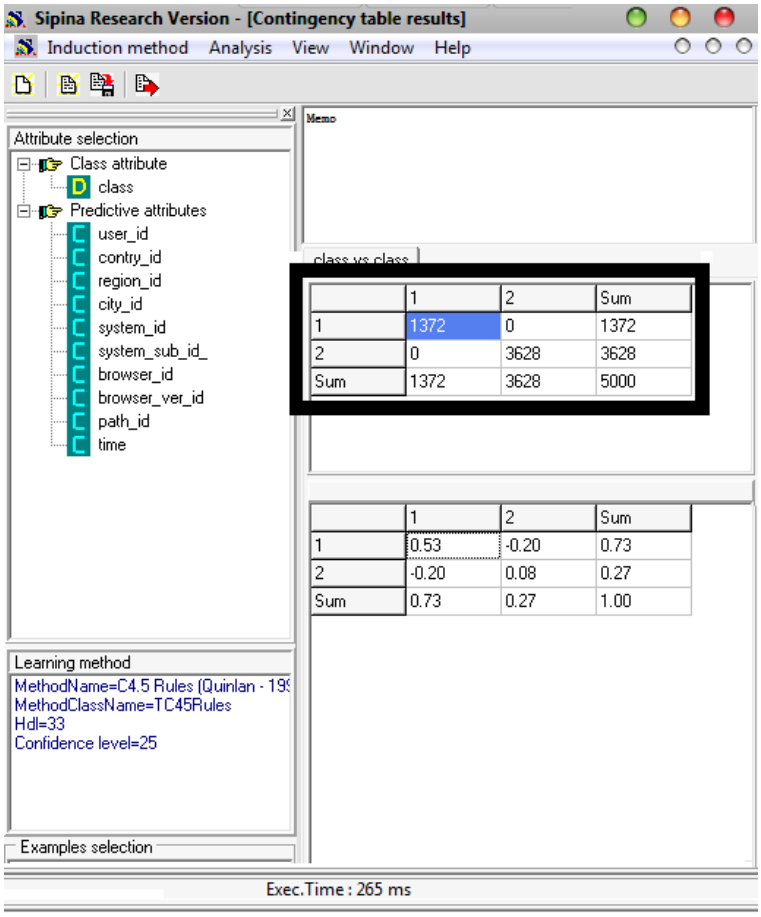
Podemos ver en la Figura 5.2, en el área resaltada con un rectángulo, los resultados que arroja esta herramienta en forma de matriz, llamada *Matriz de Confusión*, la cual será explicada más adelante de este capítulo.

5.1.3. SIPINA (Supervised Learning).

SIPINA[36] fue desarrollado también por Ricco Rakotomalala. Es un Software Libre para Minería de Datos, en donde se implementan métodos para llevar a cabo las tareas de Minería de Datos, solo que esta herramienta enfoca más su trabajo a la implementación de Árboles de Decisión, aunque también implementa otros como, Bayes Ingenuo, CN2, IREP, entre otros más. A pesar de que SIPINA es un software desarrollado por el mismo autor de TANAGRA, los métodos que se implementan presentan diferencias, en concreto métodos como Bayes Ingenuo, las implementaciones de las dos herramientas difieren una de la otra. Mostraremos también un ejemplo de

como se lleva a acabo la clasificación de un conjunto de datos, ahora considerando al método C4.5, los resultados de esta clasificación se pueden observar en la Figura 5.3:

Figura 5.3: Ejemplo de la clasificación con C4.5 de TANAGRA



5.1.4. Implementaciones propias desarrolladas en Matlab 7.7 (R2008b).

Las **Implementaciones desarrolladas en Matlab 7.7 (R2008b)**, son algunos clasificadores, que decidimos implementar dada que presentaban una exactitud mayor al momento de clasificar con los conjuntos de prueba seleccionados, entre estos se encuentran, Bayes Ingenuo, Perceptrón Multicapa, SVM, ID3, C4.5, por mencionar a algunos de los que se implementaron. Como vimos en el Capítulo 4 también se implemento la aplicación del MVBA propuesto.

Las herramientas WEKA, TANAGRA y SIPINA fueron consideradas porque su uso es relativamente fácil, se encuentran reportadas en muchos trabajos relacionados con Minería de Datos, además de que son Software Libre el cual se pueden ser descargadas desde la página de cada una de ellas. Las implementaciones que se realizaron son de gran importancia ya que así podemos experimentar con diferentes métodos de clasificación y comparar nuestros resultados con los de las demás herramientas utilizadas.

5.2. Formato de las pruebas realizadas

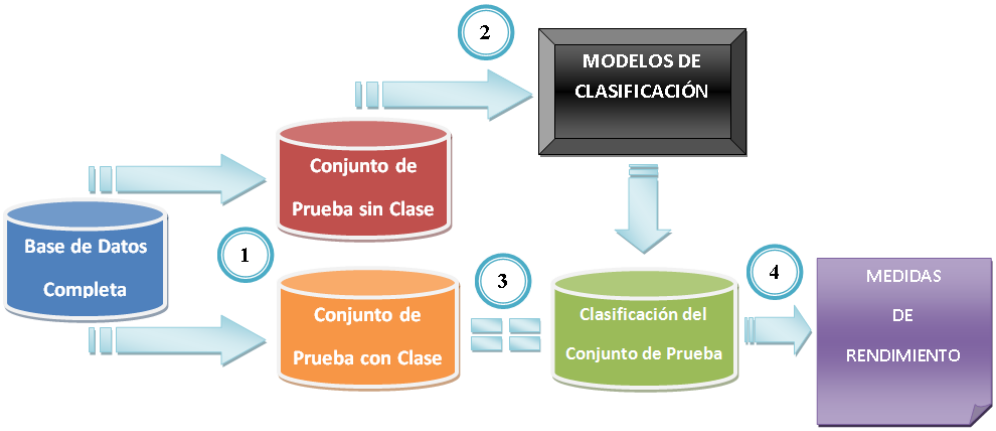
Se realizaron pruebas de clasificación con las herramientas vistas en la introducción de este capítulo, de donde seleccionamos diferentes tamaños para el conjunto de prueba, tomados de manera aleatoria, de la BD gemius_completa. Se propusieron 13 tamaños del conjunto de prueba, variando el número de registros considerados, la Tabla 5.1 muestra estos 13 tamaños que se le dio al conjunto de prueba.

Tabla 5.1: Tamaños del conjunto de prueba.

Caso	*Tamaño del conjunto de prueba*
1	5000
2	10000
3	20000
4	40000
5	80000
6	120000
7	160000
8	200000
9	240000
10	280000
11	320000
12	360000
13	BD Completa (379485)

Cada prueba que se realizaba con las diferentes herramientas consideradas seguía el esquema de funcionamiento, que se muestra en la Figura 5.4:

Figura 5.4: Esquema de funcionamiento de las pruebas realizadas.



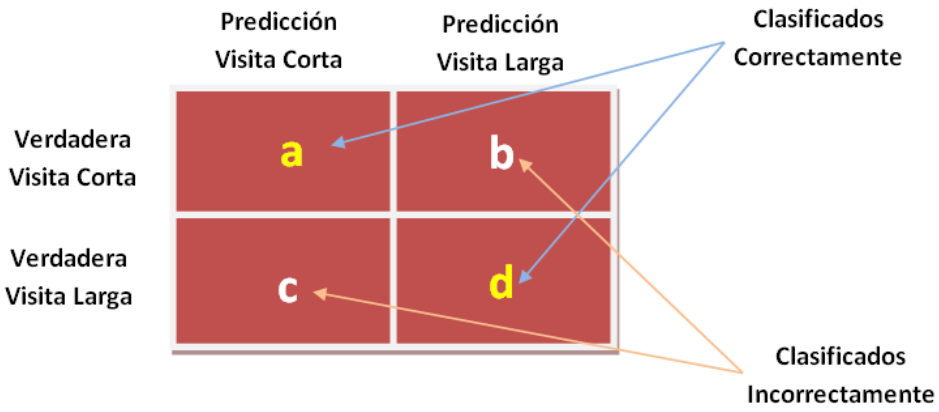
1. Como se puede ver en la Figura 5.4, se toman dos conjunto de la BD gemius_completa, un conjunto sin clase y otro con la clase de cada uno de los ejemplos, estos serán el conjunto de prueba y el de entrenamiento respectivamente para cada prueba.

2. El conjunto de prueba se aplica a cada modelo de clasificación seleccionado, para encontrar la clasificación de cada ejemplo.

3. Una vez que se obtiene las clasificación, se iguala con el conjunto de entrenamiento, para ver que tan exacta es la clasificación del método de clasificación que se elija.

4. Al final con esta comparación, se tienen estadísticas que nos permiten calcular las principales medidas de rendimiento utilizadas en Minería de Datos.

Al finalizar el proceso de prueba, debemos elegir cual va a ser la medida de rendimiento, para saber que tan bien o mal se clasifican los datos con alguno de los métodos de clasificación. La medida de rendimiento elegida fue la llamada *exactitud*, la cual nos proporciona información del porcentaje de clasificados correctamente, de entre todo el conjunto de prueba. Esta medida de rendimiento se calcula en base a una matriz llamada *Matriz de Confusión*, que se define de la siguiente manera:



Esta matriz de confusión, muestra qué ejemplos fueron clasificados correctamente (a, d) y cuales fueron clasificados de forma incorrecta (c, b), estos valores son usados para proponer la siguientes medida de rendimiento:

$$Exactitud = \frac{(a+d)}{(a+b+c+d)}$$

$$Mejora = \frac{a/\,(a+b)}{(a+c)\,/\,(a+b+c+d)}$$

$$\textit{Precisión} \;= \frac{a}{(a+c)}$$

$$Memoria = \frac{a}{(a+b)}$$

Seleccionamos exactitud como medida de rendimiento, ya que es la más utilizada, en la mayoría de los trabajos de Minería de Datos, para tener una idea de que tan bueno es un método de clasificación. Cabe mencionar que para cada una de las pruebas realizadas se ocupo la validación cruzada para determinar la validez del modelo que construimos.

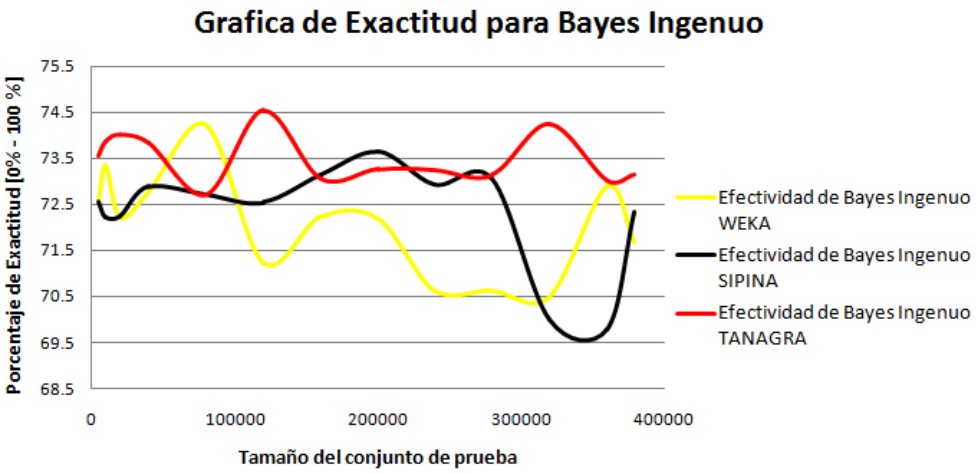
5.3. Comparación de Resultados.

Se realizaron las pruebas descritas en la sección anterior, con las diferentes implementaciones, que presenta cada una de la herramientas, de los siguientes cuatro métodos de clasificación:

1. Método de Bayes Ingenuo.
2. Métodos de Arboles de Decisión.
3. Métodos de Reglas de Decisión.
4. Método por Votación.

El primer método que presentaremos es el de Bayes Ingenuo, en donde se calculó la exactitud de las diferentes implementaciones de las herramientas consideradas. Los resultados obtenidos se presentan en la Figura 5.5:

Figura 5.5: Exactitud de Bayes Ingenuo.

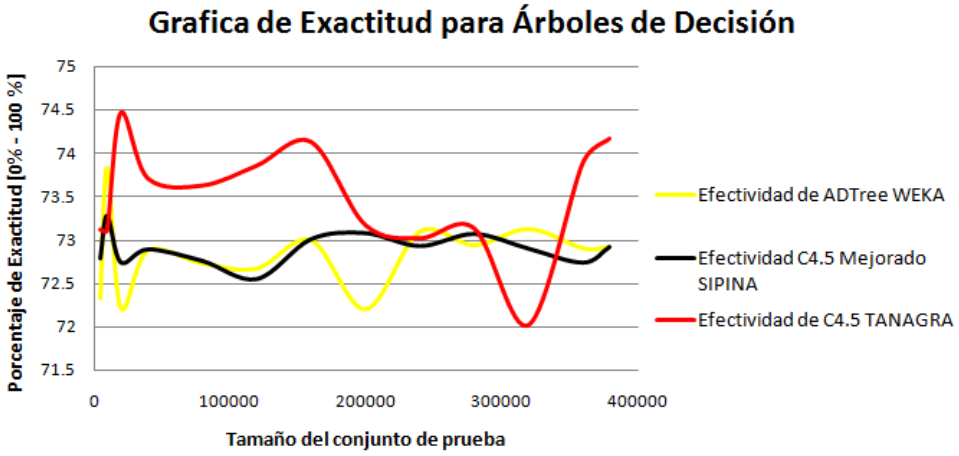


Se puede observar en la Figura 5.5, que la mejor exactitud la tiene Bayes Ingenuo de TANAGRA, ya que se adapta mejor a esta BD obteniendo una exactitud promedio del 73.52 %, en las 13

pruebas que se realizaron, esto es porque se tomaron los tamaños del conjunto de prueba vistos en la Tabla 5.1.

El segundo método que se revisó fue el de Árboles de Decisión, para este tipo de modelo de clasificación, dentro de las herramientas existe un gran número de métodos implementados, mostraremos aquellos que presentaron una mayor exactitud para las herramientas de Minería de Datos. En la Figura 5.6 se muestran los resultado obtenidos con los métodos de clasificación seleccionados.

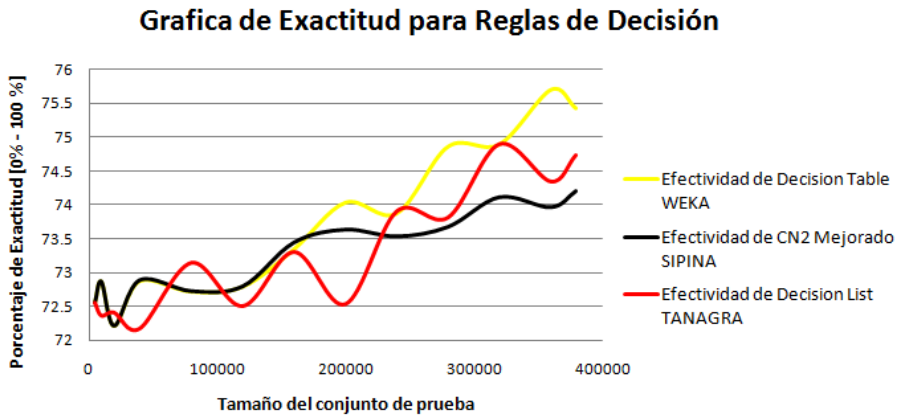
Figura 5.6: Exactitud de Arboles de Decisión.



En la Figura 5.6 podemos observar que el método C4.5 de TANAGRA, es el que muestra una mejor exactitud con respecto a C4.5 de SIPINA y ADTree de WEKA, obteniendo un 73.50 % de exactitud promedio, de las 13 pruebas realizadas.

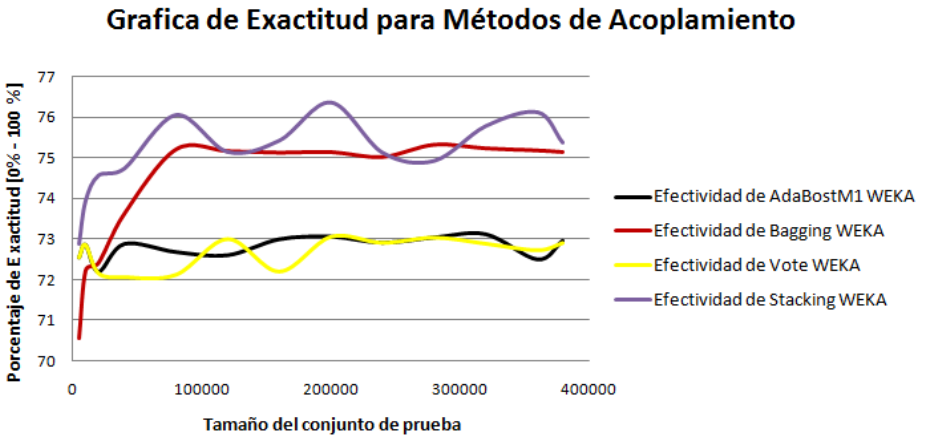
El tercer método que revisaremos es el de Reglas de Decisión, al igual que Árboles de Decisión, existen un número considerable de algoritmos implementados en las herramientas consideradas, mostraremos los resultados con las implementaciones que mostraron una mejor exactitud, las cuales se pueden ver en la Figura 5.7.

Figura 5.7: Exactitud de Reglas de Decisión.



En esta ocasión la mejor exactitud promedio la tiene Decision Tables de WEKA con un 73.71 %. El cuarto método revisado fueron los clasificadores basados en acoplamientos, de los cuales se selecciono la herramienta WEKA, ya que tenía un gran número de implementaciones que presentaban una mejor exactitud con respecto a las otras dos herramientas. Los resultados se pueden ver en la Figura 5.8.

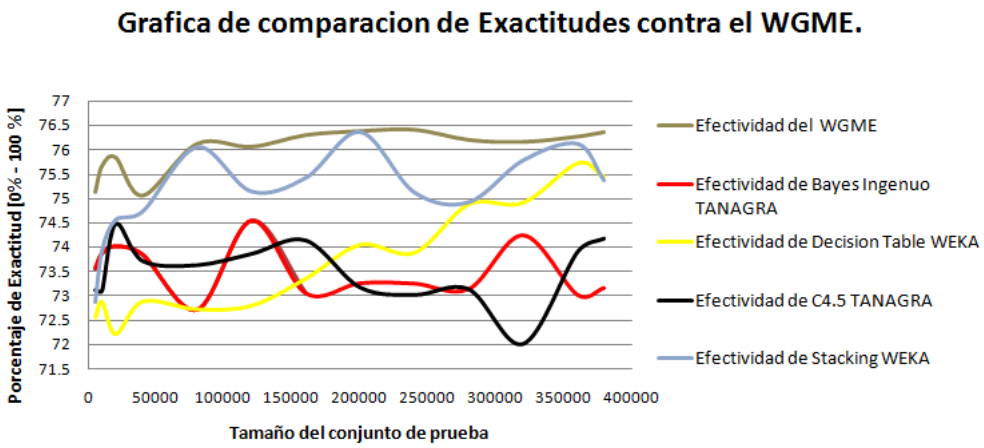
Figura 5.8: Exactitud de los Métodos por Votación.



En esta ocasión el método *Stacking* muestra la mejor exactitud, este método es similar al WGME, es un método de generalización apilada con un criterio de votación por mayoría, este método nos permite seleccionar aquellos clasificadores que van a formar parte de la clasificación, en este caso seleccionamos cuatro clasificadores Bayes Ingenuo, ADTree, Decision Tables y CitatioKNN para la Base del primer nivel de los clasificadores, en el caso del metaclasificador escogimos a Bayes Ingenuo. Con este método se obtiene 75.19 % de exactitud promedio. Resolvimos el problema de la longitud de la vista con un 73.98 % de exactitud promedio, considerando las mejores implementaciones de las herramients consideradas.

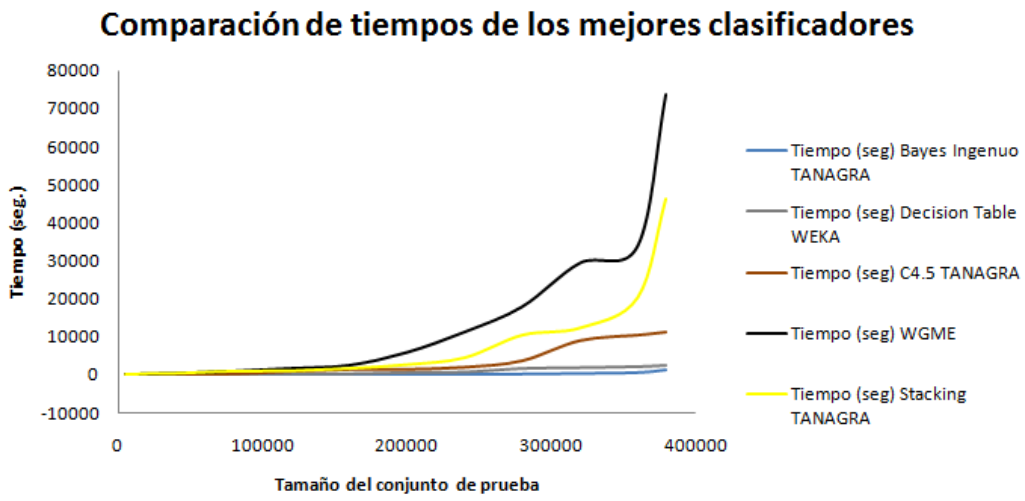
Como se observó en las figuras anteriores, en cada uno de los métodos revisados hubo una herramienta en particular que presentó los mejores resultados, los cuales tomamos para compararlos con los resultados obtenidos con el WGME, realizando cada una de las pruebas de clasificación con los conjuntos vistos en la Tabla 5.1. De esta última comparación nuestra propuesta es mejor ya que presenta un incremento promedio de exactitud del 2.13 %, respecto a los mejores métodos resultados con los cuatro métodos descritos anteriormente. Esto nos dice que nuestro método presenta mejores resultados con respecto a los herramientas tradicionales, obteniendo un 76.03 % de exactitud en promedio, tal y como se puede ver en la gráfica mostrada de la Figura 5.9:

Figura 5.9: Comparación final de resultados.



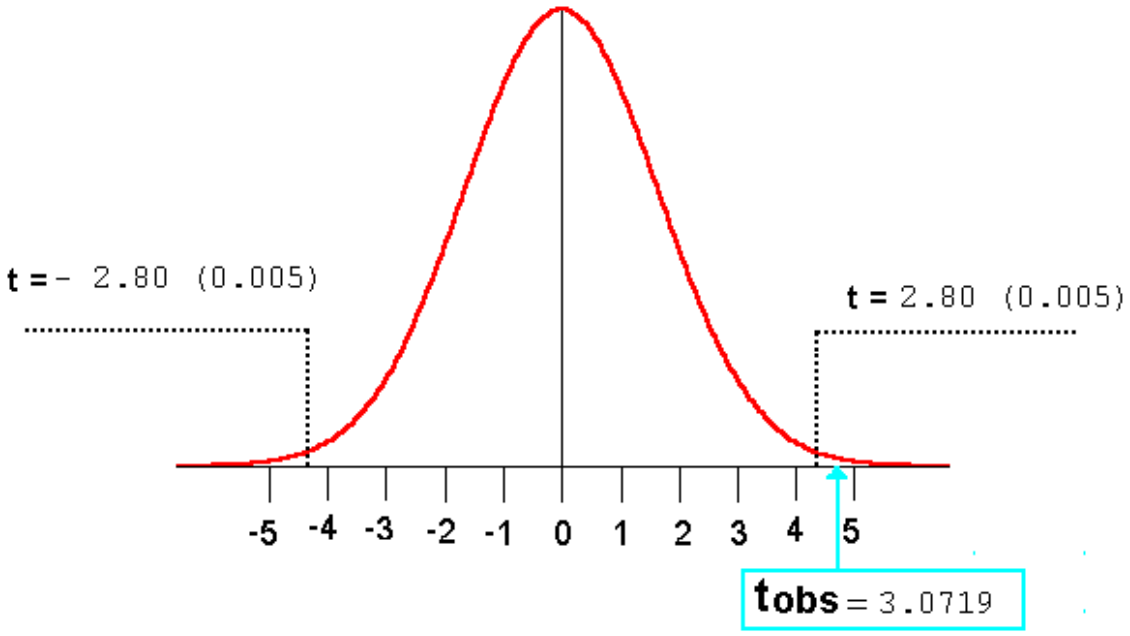
Solo queda por mostrar los tiempos de ejecución de cada uno de estos métodos, en este caso veremos que tanto Stacking de WEKA como el WGME, son los que consumen un mayor tiempo. Esto debido a que como se encuentran presentes varios clasificadores, el tiempo en el que finalizan es aproximadamente igual a la suma de los tiempos individuales de los clasificadores presentes, en promedio Stacking finaliza su clasificación en 2 horas y 16 minutos, mientras que WGME lo hace en 4 horas y 19 minutos, estos tiempos de ejecución se pueden observar en la Figura 5.10.

Figura 5.10: Comparación de tiempos de ejecución.



Estos son los resultados que obtuvimos con esta BD, a fin de la significancia estadística procedimos a calcular el t-Test, tomando como el conjunto de muestras, las diferentes exactitudes obtenidas por el WGME y Stacking de WEKA, que como ya se menciono fue seleccionado por ser el de mayor exactitud. En la sesión de pruebas para los subconjuntos presentados en la Tabla 5.1, al realizar este proceso encontramos que el valor de t = 3.0719, ahora tenemos 13 muestras por lo que n = 13 y tenemos df = 24, el cual es el valor de los grados de libertad, consultamos entonces la Tabla 3.4 del Capítulo 3, para obtener el nivel de significancia para nuestras pruebas, obteniendo la Figura 5.11:

Figura 5.11: Distribución del Muestreo de t = 4.8 para df = 12 (df = n -1).



Se puede observar en la gráfica de la Figura 5.11, que el nivel de significancia es alto, lo que tenemos es un 0.005 con un t teórico igual a 2.80, esto nos dice que estamos seguros de que los resultados son significativamente diferentes con un 99.995 %. Las medidas de rendimiento restantes, comparadas con el método *Stacking* de WEKA, se pueden observar en la Tabla 5.2, en donde *TCP* son los tamaños de los conjuntos de prueba, la *Mejora* se refiere a que tan bien es la predicción aleatoria dentro de una fracción de los datos que fueron predichos como verdaderos, *Precisión* hace referencia a cuantas clases son arrojadas como correctas y por último *Memoria* es cuantos positivos correctos son arrojados por el clasificador. Podemos ver en la Tabla 5.2 que en cada una de las medidas de rendimiento y para cada tamaño del conjunto de entrenamiento, nuestra propuesta presenta un mejor desempeño con respecto al método *Stacking*, que fue el que presentó la mejor exactitud en la fase de pruebas.

Tabla 5.2: Comparación final de medidas de rendimiento.

	WGME			*Stacking (WEKA)*		
TCP	*Mejora*	*Precisión*	*Memoria*	*Mejora*	*Precisión*	*Memoria*
5000	1.9985	0.5489	0.5328	1.9737	0.5403	0.5137
10000	2.0307	0.5509	0.5603	1.9863	0.5476	0.5458
20000	2.0100	0.5585	0.5784	1.9901	0.5123	0.5501
40000	2.0115	0.5453	0.5237	1.9787	0.5200	0.5347
80000	2.0546	0.5610	0.6246	1.9814	0.5318	0.5915
120000	2.0208	0.5534	0.5914	1.9623	0.5385	0.5901
160000	2.0480	0.5529	0.5685	1.9914	0.5437	0.5124
200000	2.0752	0.5589	0.6200	1.9817	0.5429	0.5872
240000	2.0691	0.5603	0.6133	1.9986	0.5543	0.5918
280000	2.0743	0.5537	0.6052	2.0234	0.5367	0.5872
320000	2.0671	0.5499	0.6217	1.9937	0.5253	0.6013
360000	2.0212	0.5547	0.5613	1.9765	0.5496	0.5567
379485	2.0182	0.5591	0.5663	1.9813	0.5407	0.5453

También es necesario ver como se compara nuestro método con los resultados dentro del ECML/PKDD'2007 Discovery Challenge[33]. Dembczynski et al. reportan un 75.94 % de exactitud en promedio, Hassan et al. y Lee obtienen un 76.64 % de exactitud en promedio y finalmente Mavroeidis reporta un 75.56 % de exactitud en promedio. Estos son los tres principales resultados en esta conferencia donde se utilizó la misma BD que presentamos en este trabajo. Comprarando los resultados obtenidos por el WGME se puede ver que estamos detras de los resultados de Lee con un 1 %, esto nos dice que estamos en un rango muy aceptable con respecto a los valores de exactidud que obtuvimos, con los mejores resultados obtenidos en el ECML/PKDD'2007 Discovery Challenge[33]. No podemos decir que esta comparación de resultados sea profunda, ya que no contamos con un gran número de trabajos generados en el año en que se llevo a cabo esta con-

ferencia, solo contamos con los 3 más importantes trabajos para el ECML/PKDD'2007 Discovery Challenge. Finalmente mostraremos los resultados finales que se obtuvieron con esta BD y algunas más del repositorio de la Universidad de California, Irvine[4], lo cuales se muestran en la Tabla 5.3.

Tabla 5.3: Comparación final de resultados.

		Exactitud		
Nombre	*Registros*	*WGME*	*Bayes Ingenuo (WEKA)*	*Stacking (WEKA)*
Gemius_complete	379485	76.03	74.27	75.19
Credit (German)	1000	72.33	70.03	71.75
Mushroom	8124	92.46	90.07	95.63
Australian	690	84.12	79.16	82.75

En la Tabla 5.3 se puede observar que para la BD gemius_completa, Credit (German) y Australian, la mejor exactiud la obtiene nuestra propuesta, comparando estos resultados con los dos métodos de la herramienta WEKA, que fueron los que mejores resultados obtuvieron en la parte de pruebas. Solo para el caso de la BD Mushroom, Stacking obtiene mejor exactitud que nuestra propuesta.

Capítulo 6

Conclusiones

6.1. Objetivos

Este trabajo tuvo por objetivo desarrollar un técnica novedosa para poder aplicar la tarea de clasificación de grandes cantidades de datos, tomando como referencia algunos de los métodos tradicionales que se encuentran reportados en la literatura, como son Árboles de Decisión, Reglas de Decisión, Clasificadores basados en casos, Clasificadores Bayesianos, Clasificadores por votación, entre otros más. El proceso para la obtención de la teoría y las herramientas que nos proporcionarán estos clasificadores, fue un proceso largo ya que dentro de la mayoría de los trabajos relacionados con Minería de Datos, existen un gran número de herramientas que implementan este tipo de clasificadores. Esto nos llevó a seleccionar solo algunas de las herramientas que se encuentran en la literatura así como la implementación propia de algunos de los clasificadores.

6.2. Desarrollo

Realizamos una serie de pruebas individuales con cada uno de los métodos de clasificación que se consideraron, tomando algunas de las herramientas para Minería de Datos más utilizadas, en donde seleccionamos WEKA, TANAGRA, SIPINA y evaluamos la exactitud de cada una de ellos, aplicados a un conjunto de BD, de entre las cuales, la BD gemius_completa utilizada en el EMCL PKDD 2007, que fue la que seleccionamos para presentar en este trabajo.

Uno de los primeros problemas que tuvimos en este trabajo fue el manejo de BD de gran

tamaño, ya que debíamos aplicar cada una de las fases del proceso KDD para llegar a la aplicación de fase de Minería de Datos. En concreto el principal problema al que nos enfrentamos fue el seleccionar un formato adecuado para esta BD, con el cual pudiéramos empezar a realizar las pruebas correspondientes para aplicar la fase de Minería de Datos. Este formato fue resuelto al aplicar las fases de limpieza, selección, preprocesamiento y transformación, con las cuales obtuvimos una forma final de nuestra BD las cual llamamos gemius_completa. El trato de estas BD largas es un proceso largo, lo cual provocó que solo presentáramos resultados en este trabajo de solamente una BD de este tipo, lo que nos pone como meta el trabajar con más BD del mundo real *largas* siguiendo la clasificación de la Tabla 1.1 de la Sección 1.1.

Una vez que tuvimos la forma adecuada de nuestra BD utilizada, realizamos pruebas con un conjunto de clasificadores de los cuales nos decidimos por implementar una aplicación con los clasificadores basados en acoplamientos ya que presentaban una mejor exactitud con respecto a los demás métodos de clasificación considerados. El objetivo de estos clasificadores es combinar un conjunto de clasificadores, en la literatura que por lo general son Árboles de Decisión, para lograr una mejor exactitud del método construido.

6.3. Logros

Nuestra propuesta fue novedosa ya que no consideramos solo a un tipo de clasificadores con en la mayoría de los trabajos, lo que realizamos fue combinar a distintos tipos de clasificadores, usando, Bayes Ingenuo, *K*-Medias, *k*-Vecinos más Cercanos, C4.5, Decisión Tables y ADTree.

Adicionalmente se utilizó una forma novedosa para encontrar las ponderaciones más adecuadas para cada uno de estos clasificadores al implementar un algoritmo genético, que nos permitió aplicar un criterio de votación ponderada. Este clasificador presentó un mejor rendimiento que los métodos de las herramientas para Minería de Datos considerados.

El nombre que le asignamos a este clasificador fue Algoritmo de Mezcla Genética de Expertos Ponderada (*Weighted Genetic Mixture of Experts* - WGME), tal y como lo dice el nombre se considera una serie de expertos, que en este caso son los clasificadores con mayor grado de exactitud, obtenidos al realizar la sesión de pruebas sobre la BD gemius_completa, entrenados de forma

separada por un conjunto de entrenamiento previamente seleccionado de esta BD.

En este caso se seleccionaron 6 clasificadores, de los cuales se obtienen sus clasificaciones individuales, estas se combinan mediante un criterio de votación ponderada, en donde el peso para cada clasificador son asignadas mediante el uso de un algoritmo genético, a diferencia de la mayoría de los trabajos que consideran las redes neuronales para obtener dicha ponderación. Una vez que se combinan las clasificaciones individuales mediante el criterio de votación ponderada se obtiene la clasificación final del conjunto de prueba.

En la comparación de nuestros resultados contra las herramientas para Minería de Datos, podemos apreciar en la Figura 5.9 que los resultados obtenidos por el WGME propuesto presentan un mayor grado de exactitud, con un 2.26 % en promedio, al aplicarlo sobre la BD gemius_completa, garantizando que existe una mejora con respecto a las herramientas consideradas.Por otro lado al comparar los resultados contra los resportados en el ECML/PKDD'2007 Discovery Challenge[33], nos encontramos solo por debajo del mejor resultado con mas del 1 % de exactitud.

Si bien tomando en cuenta que en este congreso se presentaron los mejores trabajos aplicados a la BD gemius_completa, nuestros resultados se encuentran dentro de un rango aceptable con respecto a los resultados presentados en el congreso. No podemos garantizar que el WGME se podrá comportar bien para todas las BD que probemos, este caso se puede ver en la Tabla 5.3, en donde para la BD Mushroom, Stacking obtiene mejor exactitud que nuestra propuesta. Esto es razonable ya que como menciona el Teorema del no hay comida gratis (*No free lunch*), no existe un algoritmo que tenga un mejor rendimiento para todos los problemas a los que sea sometido, siempre va existir un algoritmo que sea mejor al nuestro, tal y como sucede en este caso.

6.4. Trabajo Futuro

En este momento contamos con una aplicación realizada en Matlab 7.7 (2008b) que nos permite experimentar con más BD para poder poner a prueba su funcionamiento, en la cual faltan componentes por desarrollar para que pueda tener la misma funcionalidad como las herramientas tradicionales para Minería de Datos.

Al realizar este tipo de aplicaciones se tiene que pagar un costo de tiempo computacional muy

grande, ya que ahora tenemos la suma de los tiempos de ejecución de cada clasificador considerado, esto nos da como resultado un costo de tiempo computacional lineal. Esto presenta uno de los principales inconvenientes de esta aplicación, por lo que una línea de investigación que queda abierta es la incorporación algún modelo de programación paralela, que nos permita correr por separado cada uno de los clasificadores presentes en la aplicación, para poder reducir el costo del tiempo computacional al tiempo del clasificador más tardado.

Otra de las líneas de investigación que quedan abiertas es la del desarrollo de una herramienta más completa, agregándole las demás tareas de Minería de Datos que son, predicción, identificación, agrupamiento y asociación, así como la implementación de más clasificadores que puedan ser utilizado para el WGME. En este momento solo probamos para una sola BD del mundo real, otro de los trabajos a futuro es realizar más pruebas con este tipo de BD, como las proporcionadas por el EMCL PKDD 2008[9] y el 2009[8], para corroborar el correcto desempeño de esta aplicación a este tipo de BD.

Referencias

[1] Bauer, E., Kohavi, R.: An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants. *Machine Learning.* Vol. 36:105–139. 1999.

[2] Béjar, J.: Apuntes de Aprendizaje Automático. Universitad Politécnica de Catalunya. 2006.

[3] Bentley, J.: K-d trees for semidynamic point sets. *Annual Symposium on Computational Geometry; Proceedings of the sixth annual symposium on Computational geometry*:187–197. 1990.

[4] Blake, C., Merz, C.: *UCI repository of machine learning databases.* http://www.ics.uci.edu/m̃learn/MLRepository.html. 1998.

[5] Breiman, L.: Bagging Predictors. *Machine Learning.* Vol. 24:123–140. 1996.

[6] Brin, S.: Bagging Predictors. Near Neighbor Search in Large Metric Spaces. *The VLDB Journal*:574–584. 1995.

[7] Calders, T., Goethals, B., Prado, A.: Integrating pattern mining in relational databases. *Knowledge Discovery in Databases: PKDD 2006.* Vol. 4213:454–461. 2006.

[8] ECML/PKDD 2008.: `http://www.ecmlpkdd2009.net/`

[9] ECML/PKDD 2008.: `http://www.ecmlpkdd2008.net/`

[10] ECML/PKDD 2007.: `http://www.ecmlpkdd2007.net/`

[11] Efron, B., Tibshirani, R.: An Introduction to the Bootstrap. *Chapman & Hall*. 1993.

[12] Falcon Fraud Manager.: `http://www.fico.com/en/Pages/default.aspx`

[13] Freund, Y., Schapire, R.: A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Sciences.* Vol. 55:119–139. 1995.

[14] Fukunaga K. , Narendra, P.: A branch and bound algorithm for computing k-nearest neighbors. *IEEE Transactions on Computers.* 1975.

[15] Galbiati, J.: Distribución F de Snedecor. `http://www.jorgegalbiati.cl/nuevo\_06/Fsned.pdf`. 1990.

[16] Gemius Company.: `http://www.gemius.com/`

[17] Holland, J.: Adaptation in Natural and Artificial Systems. *University of Michigan Press, Ann Arbor.* 1975.

[18] Huber, P.: Issues in computational data analysis. *Computational Statistics. Vol. 2(Y. Dodge and J. Whittaker, eds.), Physica Verlag, Heidelberg.* 1992.

[19] Janez, D.: Statistical Comparisons of Classifiers over Multiple Data Sets. *Journal of Machine Learning Research* 7:1–30. 2006.

[20] Jang, J., Sun, C., Mizutani: Neuro-Fuzzy and Soft Computing. *A Computational Approach to Learning and Machine Intelligence.* McGraw Hill. 1999.

[21] Koltchinskii, V., Panchenko, D., Lozano, F.: Further explanation of the effectiveness of voting methods: The game between margins and weights. *In Proceedings 14th Annual Conference on Computational Learning Theory and 5th European Conference on Computational Learning Theory.* 2001.

[22] Larrañaga, P., Inza, I., Moujahid, A.: Tema 6. Clasificadores Bayesianos. *Universidad del País Vasco-Euskal Herriko Unibertsitatea.* 2007.

[23] Marín, D.: Tema 5: Análisis de Cluster y Multidimensional Scaling. *Universidad Carlos III de Madrid.* 2001.

[24] Mark, H., Eibe, F.: Combining Naive Bayes and Decision Tables. *Proc 21st Florida Artificial Intelligence Research Society Conference.* 2008.

[25] McClelland, J., Rumelha9t, D.: Learning, Representations by back-propagation. *Nature.* 1986.

[26] Mitchell, T. M.: Machine Learning. *McGraw-Hill Science/Engineering/Math.* 1997.

[27] Polikar, R.: Ensemble Based Systems in Decision Making. *IEEE Circuits and Systems Magazine.* Vol. 6:21–45. 2006.

[28] Quinlan, J.: C4.5: Programs for Machine Learning (Morgan Kaufmann Series in Machine Learning). *Morgan Kaufmann.* 1993.

[29] Quinlan, J.: Induction of decision trees. *Machine Learning.* Vol. 1:81–106. 1986.

[30] Rosenblatt, F.: The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. *Cornell Aeronautical Laboratory, Psychological Review.* Vol. 65:386–408. 1958.

[31] Shannon, C.: A mathematical theory of communication. *Bell System Technical Journal.* Vol. 27:379–423 y 623–656. 1948.

[32] Schapire, R.: The boosting approach to machine learning: An overview. *AT&T Labs Research Shannon Laboratory.* 2001.

[33] The 18th European Conference on Machine Learning and The 11th European Conference on Principles and Practice of Knowledge Discovery in DataBases.: *Proceedings of the ECML/PKDD'2007 Discovery Challenge.* 2007.

[34] Universidad Autónoma Metropolitana.: `http://www.egresados.uam.mx/est9803/est\_9803.pdf`

[35] Universidad Autónoma Metropolitana.: `http://www.egresados.uam.mx/est\_emp\_97\_02/index.html`

[36] Witten, I., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques. *Morgan Kaufmann Publishers. 2nd edition:* 560. 2005.

[37] Wolpert, D.: Stacked generalization. *Neural Networks.* Vol. 5, no. 2:241–259. 1992.

[38] Wu, X., Kumar, V., Quinlan, J., Ghosh, J., Yang Q., Motoda H., McLachlan G., Ng, A., Liu, B., Yu, P., Zhou, ZH., Steinbach, M., Hand, D., Steinberg, D.: Top 10 algorithms in data mining. *Knowledge and Information Systems.* Vol. 14: 1–37. 2008.

[39] Yianilos: Data Structures and Algorithms for Nearest Neighbor Search in General Metric Spaces. *SODA: ACM-SIAM. Symposium on Discrete Algorithms (A Conference on Theoretical and Experimental Analysis of Discrete Algorithms).* 1993.