Departamento de Computación

Facultad de Ciencias Exactas y Naturales – Universidad de Buenos Aires

-Tesis de Licenciatura-

Perfiles de Usuario para Agentes de Interfaz: Un análisis de técnicas de aprendizaje

Autor
Valerio Adrián Anacleto
vanaclet@dc.uba.ar
Universidad de Buenos Aires
Capital Federal, Argentina

Director
Dra. Analía Amandi
amandi@exa.unicen.edu.ar
Universidad Nacional del Centro

Co-Director
Ms.CS. Marisa Bauza
mbauza@dc.uba.ar
Universidad de Buenos Aires
Capital Federal, Argentina

abril de 2003

Resumen

En este trabajo mostraremos la aplicación de técnicas complejas de aprendizaje de máquina aplicadas a la resolución de problemas de aprendizaje por parte de agentes de interfaz autónomos.
Los Agentes de Interfaz Autónomos [AIA/1997] asisten a usuarios en la realización de tareas cotidianas, intentando actuar como un secretario humano lo haría. Para ello debe conocer al humano al que asiste, pudiendo, de esta manera, cumplir su tarea en forma eficiente. Para conocer al usuario, el agente deberá aprender los gustos de éste. En nuestro caso, el aprendizaje se realiza netamente, a través de la observación del uso del sistema por parte del usuario, sin consultas del agente, ni respuestas directas del usuario.
El entorno en que se desempeña el agente es un sitio de comercio electrónico. Una de sus tareas es observar los hábitos y acciones del usuario, tomando esta información dinámicamente. La información recogida será utilizada para inferir cuáles son las acciones que el usuario seguirá en su recorrido por el sitio. Este tipo de análisis y colecta de datos dinámicos, posee una función objetivo difusa y, en la que, en general, no se cuenta con datos previos para realizar el entrenamiento. Este factor hace que técnicas de data mining y machine learning clásicas deban ser reconsideradas, quizás fallen, o quizás deban ser descartadas.
En particular, un problema importante es resolver cómo manejar la amplia cantidad de datos que se pueden tomar de manera directa, a partir de las acciones del usuario, aun de manera fuera de línea[KOH/2001]. Este problema se ve ampliado si queremos tomar estos datos dinámicamente, procesarlos y, darle una respuesta al usuario de manera "on line".
A lo largo del presente trabajo se implementan y analizan tres técnicas de aprendizaje para resolver este problema: algoritmos genéticos, redes bayesianas, y redes neuronales. Los resultados obtenidos son mostrados de manera gráfica, analizando también, su posible utilización en otros dominios. Buscamos con este trabajo traer luz al problema de evaluar qué técnicas permiten que los Agentes de Interfaz actúen más eficientemente, en qué dominios se desempeñan mejor y con qué parámetros, en un entorno dinámico como es el comercio electrónico.

Agradecimientos

A mi novia, "Vivi", por 7 años de paciencia a toda prueba.
A mi familia; en especial a mi viejo que seguramente me brinda su ayuda, desde donde esté.
A mi abuela, "Chala", por estar en todo momento.
A mi directora, Analía Amandi, por su excelencia académica y docente, las cuales hacen que sea un orgullo para mí haber terminado este trabajo junto a ella.
A mi co-directora, Marisa Bauza, por sus sabios comentarios, que sumados al apoyo moral, facilitaron la finalización de esta tesis.
A mis amigos.
Finalmente, aunque no menos importante, a todos los docentes que diariamente se levantan pensando cómo hacer mejor su trabajo, y que luchan por la universidad pública y gratuita a pesar de todo.

Índice

1. Introducción ____ *17*

1.1. Motivación ____ **17**

1.2. Contribuciones ____ **19**

1.2.1. Aplicar Técnicas Complejas de Inteligencia Artificial al Problema de Generación de Perfiles 19

1.2.2. Contratación de las Técnicas ____ 20

1.2.3. Un Framework Para Medir la Performance de Distintas Implementaciones ____ 20

1.3. Estructura de la Tesis ____ **20**

2. Agentes ____ *23*

2.1. Perfiles de Usuarios Para Conocer los Gustos de Nuestros Clientes ____ **24**

2.2. Técnicas de Construcción de Perfiles ____ **24**

2.2.1. Content Based ____ 25

2.2.2. Collaborative ____ 26

2.2.3. Técnicas de Construcción de Perfiles Dinámicos ____ 26

2.3. Aprender Perfiles Como un Problema de Aprendizaje de Máquina ____ **27**

2.3.1. Aprendizaje de Máquina ____ 27

2.3.2. Definición de Aprendizaje ____ 27

2.3.3. Elección de Técnicas de Aprendizaje ____ 28

2.4. Agentes de Software para Generar Perfiles Dinámicos de Usuarios ____ **29**

2.4.1. Agentes de Software ____ 29

2.4.2. Historia de los Agentes ____ 29

2.4.3. Características Principales de los Agentes ____ 30

2.4.4. Clasificación de Agentes ____ 31

2.5. Agentes Interfaz Para Aprender Perfiles de Usuarios ____ **33**

2.5.1. Comportamiento de Compra ____ 34

2.5.2. Como se Relacionan los Agentes con el CBB ____ 34

2.5.3. Algunas Aplicaciones de Agentes y su Utilización. ____ 34

2.6. Comentarios Finales ____ **37**

3. Vendor un Agente Para Comercio Electrónico ____ *39*

3.1. Vendor ____ **39**

3.1.1. Vendor - Capacidades ____ 40

3.2. Construcción del Perfil de Usuario ____ **41**

3.2.1. Observación ____ 41

3.2.2. Casos ____ 42

3.2.3. Aprendizaje ____ 44

3.3. Asistencia al Usuario ____ **47**

3.4. Registro Histórico ____ **48**

3.5. Independencia de la Técnica de Aprendizaje ____ **48**

3.6. Comentario Finales ____ **48**

4. Perfiles Utilizando Redes Bayesianas ____ *51*

4.1. Redes Bayesianas ____ **51**

4.2. Representación de Conocimiento 52
4.2.1. Semántica de una Red Bayesiana 52
4.2.2. Como Inferir un Determinado Valor 55
4.2.3. Tipos de Problemas 56

4.3. Redes Bayesianas Adaptativas 56

4.4. Propiedades de las Redes Bayesianas 57
4.4.1. Sesgo Inductivo 57
4.4.2. Problemas Apropiados 57
4.4.3. Ruido 57
4.4.4. Generalización de Nuevos Casos 57
4.4.5. Sobreaprendizaje 57

4.5. Redes Bayesianas Para Generar Perfiles Dinámicos de Usuarios 58
4.5.1. Representación de Perfiles por Medio de una Red Bayesiana 58

4.6. Arquitectura de la red 59

4.7. Modo de Actualización 60
4.7.1. Ejemplos 61
4.7.2. Algoritmo 66

4.8. Comentarios Finales 68

5. Perfiles Utilizando Algoritmos Genéticos *69*

5.1. Algoritmos Genéticos 70

5.2. Anatomía de un Algoritmo Genético Simple 70
5.2.1. Codificación de los Cromosomas 71
5.2.2. Evaluación y Selección 72
5.2.3. Operadores Genéticos 74

5.3. Propiedades de los Algoritmos Genéticos 77
5.3.1. Lenguaje de Representación 77
5.3.2. Sesgo 78
5.3.3. Tipos de Problemas Apropiados 78
5.3.4. Comportamientos Comunes: 78

5.4. Algoritmos Genéticos Para Generar Perfiles Dinámicos de Usuarios 79
5.4.1. Representación de Perfiles Utilizando Algoritmos Genéticos 79

5.5. Generando Perfiles Dinámicos de Usuarios Utilizando Algoritmos Genéticos 86
5.5.1. Parámetros del Algoritmo 86

5.6. Aprender Perfiles por Medio de Algoritmos Genéticos, Enfoques Posibles Para Resolver Este Problema 88
5.6.1. Implementación I 89
5.6.2. Implementación II 91
5.6.3. Implementación III y la Utilización de Memoria 93
5.6.4. Implementación IV 95
5.6.5. Implementación V la Versión Final 96
5.6.6. Conclusión 98

5.7. Comentarios Finales 98

6. Perfiles Utilizando Redes Neuronales *99*

6.1. Redes Neuronales de Tipo Biológico 99

6.2. Redes Neuronales Artificiales 100
6.2.1. Estructura y Formas de Interconexión 102
6.2.2. La Función de Propagación (de red o de base) 103
6.2.3. La Función de Activación 103

6.3. Caracteristicas y Modo de Operación de las Redes Neuronales 105
6.3.1. Modo de Operación y Tipos de Aprendizaje 105
6.3.2. Aprendizaje - Redes de Retropropagación 109
6.3.3. Propiedades de las Redes Neuronales 112

6.4. Redes Neuronales Para Generar Perfiles Dinámicos de Usuarios 114
6.4.1. Representación de Perfiles por Medio de una Red Neuronal 114
6.4.2. Aprendizaje y Diagnóstico 119

6.5. Implementación 120
6.5.1. Codificación de las Variables 121
6.5.2. Arquitectura de la Red 122

6.6. Comentarios Finales 124

7. Resultados Obtenidos *125*

7.1. Refinamiento del Perfil de Usuario a lo Largo del Tiempo 126
7.1.1. Análisis de Refinamiento y Mejora del Perfil a lo Largo del Tiempo 127
7.1.2. Análisis de Refinamiento a lo Largo del Tiempo de Conceptos Particulares 132

7.2. Adaptación a los Intereses Dinámicos del Usuario 135
7.2.1. Experimentos Realizados 135

7.3. Análisis de Tiempo y Memoria 137

7.4. Comentarios Finales 139

8. Cometarios Finales *141*

8.1. Resumen 141

8.2. Trabajo Futuro 144

A. Análisis Estadístico de los Datos *147*

A.1. Conceptos Teóricos 147
A.1.1. Dominio 147
A.1.2. Datos Desconocidos 148
A.1.3. Ruido 148

A.2. Caracterización del Dominio 149
A.2.1. Nuestro Dominio 149

A.3. Características generales del concepto a aprender 150
A.3.1. Relevancia presunta de cada atributo 150

A.4. Cometarios Finales 156

B. Un Framework Para Implementar Agentes de Interfaz en Entornos de Comercio Electrónico *157*

B.1. Diseño e Implementación 158

B.2. Descripción de las clases 159

B.3. Cometarios Finales 160

Índice de Figuras

Figura 2.1 - Diagrama de un esquema de perfiles Content Based 25
Figura 2.2 - Diagrama de un esquema de perfiles Collaborative..... 26
Figura 2.3 - Diagrama de Perfiles por usuario (Content Based)..... 27
Figura 2.4 - Topología de Agentes..... 32
Figura 3.1 - Vista de la Barra de navegación de la aplicación 43
Figura 3.2 - Ejemplo de un caso armado por el agente, en base a una acción de búsqueda del usuario..... 43
Figura 3.3 - Vista de una pantalla de la aplicación 45
Figura 3.4 - Caso de correspondiente a la pantalla de la figura 3.3..... 46
Figura 3.5 - Funcionalidad completa de Vendor 49
Figura 4.1 - Ejemplo de una red bayesiana sencilla..... 53
Figura 4.2 - Probabilidad para el ejemplo de la Alarma..... 55
Figura 4.3 - Cálculo de una entrada de probabilidad..... 55
Figura 4.4 - Diseño de la red bayesiana utilizada por Vendor 60
Figura 4.5 - Caso a aprender por la red bayesiana 62
Figura 4.6 - Red observada con los valores de los atributos del caso 63
Figura 4.7 - Caso a aprender por la red bayesiana 64
Figura 4.8 - Algoritmo de actualización -Primera iteración..... 65
Figura 4.9 - Algoritmo de actualización -Segunda iteración 65
Figura 4.10 – Pseudocódigo del algoritmo de actualización 67
Figura 5.1 - Esqueleto de un algoritmo genético 71
Figura 5.2 - Ejemplos crossover..... 75
Figura 5.3 – representación de un Individuo..... 80
Figura 5.4 – Ejemplo Crossover 81
Figura 5.5 – Ejemplo de mutación 82
Figura 5.6 – Ejemplo - agregar un gen 83
Figura 5.7 – Caso con el que se medirá el fitness de un individuo 85
Figura 6.1 - Neuronas e interconexión..... 100
Figura 6.2 - Unidad de proceso típica..... 101
Figura 6.3 - Modelo de red en cascada de 3 capas 102
Figura 6.4 - Perceptrón de salida binaria..... 107
Figura 6.5 - Calculo de entrada ponderada del nodo Yj 110
Figura 6.6 – Efecto de una capa oculta sobre las unidades de salida..... 111
Figura 6.7 - Ejemplo de codificación correspondiente a la primer capa de la red neuronal 116
Figura 6.8 – Primeras dos capas de la red neuronal..... 117
Figura 6.9 – Red neuronal utilizada para aprender un perfil de usuario con dos atributos 119
Figura 6.10 – Algoritmo utilizado para el aprendizaje de nuestra implementación utilizando redes neuronales 120
Figura 6.11 - Estructura de la red neuronal para representar perfiles de usuarios 123
Figura 7.1 - Adaptación del usuario "Viajante" 129
Figura 7.2 - Adaptación del usuario "Futbolista" 130

Figura 7.3 - Adaptación del usuario "Utilero" 131
Figura 7.4 - Comprador de ofertas y artículos de segunda 132
Figura 7.5 - De Futbolista a Viajante 136
Figura 7.6 - De Viajante a Futbolista 137
Figura 7.7- Comparación memoria consumida 138
Figura 7.8 - Comparación tiempo consumido 139
Figura 7.9 - Cuadro comparativo 140
Figura A.1 - Distribución de los Atributos 154
Figura B.1 – Arquitectura del Agente 158

Índice de Tablas

Tabla 3.1 - Atributos Involucrados en el dominio de la aplicación.....42
Tabla 4.1 - Tabla de probabilidad condicional para la variable alarma.....54
Tabla 4.2 - Probabilidad inicial de los elementos de la variable *target*.....61
Tabla 4.3 - Probabilidad final de los elementos de la variable *target* 61
Tabla 5.1 - Ejemplo de evaluación y selección 73
Tabla 5.2 - Otros Parámetros del algoritmo genético.....88
Tabla 5.3 - Variables del dominio 89
Tabla 6.1 - Comparación entre las neuronas biológicas reales y las unidades de proceso artificiales 102
Tabla 6.2 – Subconjunto de variables usado en el ejemplo de aprender un perfil, utilizando redes neuronales. 115
Tabla 6.3 - Codificación de las variables del perfil 122
Tabla 7.1 - Comparación tiempo consumido.....127
Tabla A.2 - Distribución y valores de los atributos del dominio.....154
Tabla A.3 - Correlación Marca con Categoría de producto.....155
Tabla A.4 - Correlación Categoría Comprador con Deporte 156
Tabla A.5 - Correlación Precio con Exposición 156

Capítulo I

1. Introducción

Los Agentes de Interfaz Autónomos [AIA/1997] asisten a usuarios en la realización de tareas, intentando actuar como un secretario humano lo haría. Para ello debe conocer al humano al que asiste, para así cumplir su tarea en forma eficiente.
Técnicas de inteligencia artificial han sido utilizadas para el aprendizaje de los intereses del usuario a partir de la observación del mismo. En este trabajo se analizarán las ventajas y desventajas de la utilización de diferentes técnicas de aprendizaje en la generación de perfiles de usuarios por parte de agentes de interfaz, considerando las complejas relaciones de dependencia entre los conceptos y atributos que conforman un perfil de usuario.

1.1. Motivación

Las aplicaciones que tratan con clientes usualmente no discriminan las distintas características que pueden presentar éstos, tratando por igual personas que no necesariamente tienen las mismas preferencias.
En aplicaciones de comercio electrónico, que generan cada vez más transacciones a través de Internet, se evidencian problemas por esta despersonalisación del sistema, ya que todas las ofertas y propuestas llegan a todos los usuarios

produciendo problemas de sobrecarga de información para los usuarios de Internet.
Por lo tanto, para no desalentar a un posible cliente del uso de un sistema de comercio electrónico los agentes de interfaz pueden asistir a los clientes, considerando fuertemente aquella información que efectivamente le interesa. [MAE/1994][KOH/2001] Ahora bien, adquirir información sobre las preferencias de los usuarios se puede llevar a cabo a través de dos caminos: Consultando explícitamente al usuario u observando su comportamiento con el sistema.
Obtener información de las preferencias del usuario por consulta es, en la mayoría de los casos, tedioso para el usuario, con varios formularios para llenar, con datos que la mayoría de los usuarios se resisten a dar. Por esta razón, la observación del usuario utilizando el sistema es la única alternativa que consideramos viable para extraer dicha información. Mediante la observación, estos agentes tienen que deducir preferencias y gustos en el contexto en el que el cliente los utiliza usualmente.
Es evidente que resolver este problema redunda en usuarios más satisfechos en términos de menor pérdida de tiempo en análisis de ofertas y propuestas que usualmente no aceptaría, debido a que estos usuarios reciben aquellas ofertas y propuestas que tienen alta probabilidad de ser de su interés, con lo cual aumenta de manera directa las probabilidades de compra.
Como solución computacional a este tipo de problemas es que surgen los agentes de interfaz. Un agente de interfaz, es un programa que asiste a un usuario humano, sugiriéndole caminos de acción y, en algunas ocasiones, actuando en su lugar.
Para poder ayudar al usuario el agente necesita de una cierta cantidad de conocimiento sobre la tarea que debe realizar, además de conocer los intereses, hábitos y preferencias del usuario, los cuales pueden cambiar con el transcurso del tiempo.
La idea es que el agente sea capaz de adquirir conocimiento para cumplir con las expectativas del usuario. En este trabajo se construirá un agente que deberá adaptarse dinámicamente a los gustos del usuario en un ambiente de comercio electrónico, específicamente una tienda virtual de venta de artículos deportivos.
Nuestro agente para comercio electrónico deberá conocer los intereses en relación con las compras de su usuario asociado y, a partir de este conocimiento, sugerirle la realización de algunas de las compras que considere con alta probabilidad de ser de su interés.
La recepción de estas sugerencias y la realización o no de las compras, conforman otra fuente de aprendizaje, ya que permiten que el agente pueda testear sus hipótesis con relación a los intereses del usuario.
Para ir conociendo al usuario, el agente observa su comportamiento en la navegación de las páginas del sitio que contienen productos, en las compras mismas y, en la recepción de sus sugerencias. Con esta información genera un perfil que irá refinando con el tiempo. Para materializar este comportamiento, varias técnicas de inteligencia artificial han sido utilizadas. En este trabajo, se evaluarán tres de ellas: redes bayesianas, algoritmos genéticos, y redes neuronales.

1.2. Contribuciones

Los agentes de interfaz construidos en la última década han utilizado diferentes técnicas para construir perfiles de sus clientes o usuarios, desde simples reglas obtenidas a partir de la aplicación de algún algoritmo de clustering, hasta la aplicación de técnicas de aprendizaje como redes bayesianas, algoritmos genéticos o redes neuronales. Los algoritmos más simples permiten sólo encontrar patrones en un, usualmente pequeño, conjunto de datos, y los más complejos, permiten que los agentes puedan actuar con mayor información sobre las relaciones y dependencias entre los datos.
Sabemos que es una tarea pendiente, realizar una cantidad de experimentos que permitan evaluar qué técnicas permiten que los Agentes de Interfaz actúen más eficientemente, en qué dominios se desempeñan mejor y, con qué parámetros. [HYA/1999] Un entorno dinámico como el comercio electrónico [KOH/2001] se presenta como una opción rica en variantes para realizar estos experimentos. Por otro lado a pesar de que existe una cantidad importante de agentes interfaz implementados tanto en el ambiente académico como comercial, no hay aún ningún estudio comparativo sobre técnicas de aprendizaje utilizadas.
Con respecto a estas cuestiones el presente trabajo realiza los siguientes aportes.

1.2.1. Aplicar Técnicas Complejas de Inteligencia Artificial al Problema de Generación de Perfiles

Se aplicarán tres técnicas distintas para la generación de los perfiles de usuario, estas son, redes bayesianas, algoritmos genéticos, y redes neuronales. Estas técnicas fueron seleccionadas debido que en las experiencias reportadas, se presentan ventajas con relación a la obtención de perfiles con mayores detalles de relaciones entre datos, necesarios para que agentes de interfaz puedan actuar más eficientemente.
Dos grupos de métodos no son considerados en nuestro análisis. Un grupo es el formado por métodos que permiten formar clusters y, en general construir reglas simples a partir de estos. Este grupo no fue considerado, ya que sólo es aplicado para agentes con una funcionalidad trivial y, suficientemente explorada. El segundo grupo que no es considerado es aquel que está formado por algoritmos de análisis de datos que requieren grandes volumen de información para ser efectivos, lo que no ocurre en el caso de agentes de interfaz que tienen que comenzar a construir perfiles desde el comienzo del uso del sistema, contando obviamente con muy pocos datos.
Con relación a las tres técnicas que se analizarán, las implementaciones no se realizarán desde cero si no que se adaptarán herramientas existentes, como las que pueden encontrarse en [JAV/2001], [GRO/2001], [GAU/2001], [NEU/2001], y [ECJ/2001].

1.2.2. Contratación de las Técnicas

Evaluación de resultados de las distintas aplicaciones. Se mide la performance y eficacia de cada una de ellas, evaluando sus resultados en un sistema real de comercio electrónico con un conjunto de usuarios definidos con diferentes intereses. El dominio específico de esta aplicación es una tienda virtual de artículos deportivos.

1.2.3. Un Framework Para Medir la Performance de Distintas Implementaciones

La implementación se realizó de manera tal que permite la parametrización de la técnica a utilizar, de esta forma diferentes técnicas pueden ser utilizadas por el agente sin modificar la implementación del mismo. Permitiendo trabajos futuros implementando otras técnicas de aprendizaje, o combinaciones de éstas.

1.3. Estructura de la Tesis

A continuación se detalla la estructura de este trabajo, y un breve resumen de los temas que abordaremos en cada uno de los capítulos siguientes.

Capítulo II - Agentes

En donde se brindan los conceptos teóricos básicos, para entender el presente trabajo. En este capítulo se analizará qué es un perfil, cómo se lo construye y mantiene. Se verá también qué son y cómo se clasifican los agentes de software. Finalmente, veremos cómo pueden contribuir los agentes en el problema de la generación de perfiles dinámicos de usuarios.

Capítulo III – Arquitectura y Generación de Perfiles

En este capítulo se presenta la arquitectura de nuestro agente. Analizaremos cuáles son sus tareas específicas y cómo las realiza. También veremos cómo fue implementado y su interacción con el ambiente.

Capítulo IV – Redes Bayesianas

En este capítulo se explica la implementación de nuestro agente utilizando redes bayesianas. Para comprender la implementación en la primera parte se realiza una introducción a las redes bayesianas, en ésta se explican los conceptos necesarios para entender qué es y cómo se implementa una red bayesiana, quien reúna estos conocimientos puede saltearse esta parte y comenzar por la segunda sección en la que se detalla la implementación de la red. Finalmente mostramos un pequeño ejemplo para ver el funcionamiento de la red; luego se presentan las conclusiones de este capítulo.

Capítulo V – Algoritmos Genéticos

En este capítulo se explica la implementación de nuestro agente utilizando algoritmos genéticos. Al igual que en el capítulo de redes bayesianas, este capítulo está dividido en dos partes, la primera una introducción sobre el tema algoritmos genéticos, en el cual se brindan las nociones mínimas necesarias para entender la implementación, y se explica cómo esta técnica puede ser útil para aprender un perfil dinámico de usuarios para entornos de comercio electrónico. Luego se presentan una serie de posibles configuraciones e implementaciones utilizando este algoritmo.

Capítulo VI – Redes Neuronales

En este capítulo se explica la implementación de nuestro agente utilizando redes neuronales. Al igual que en los anteriores capítulos, la primera parte de éste se dedica a la introducción teórica sobre el tema redes neuronales, y cómo estas pueden contribuir a resolver nuestro problema de perfiles de usuarios dinámicos. Luego analizamos la manera en que implementamos esta técnica.

Capítulo VII – Casos de Test

En este capítulo analizaremos el desempeño de las distintas implementaciones de nuestro agente. Consideraremos relativos al desempeño factores como, Tiempos de respuesta, memoria consumida, adaptación a los gustos del usuario, y velocidad de aprendizaje. Explicaremos porque se dieron los resultados que se muestran; y analizaremos la utilidad particular de cada técnica para implementaciones del agente en nuevos dominios.

Capítulo VIII – Conclusiones

En el capítulo final expresaremos una serie de conclusiones acerca del trabajo realizado en esta tesis, daremos un resumen de la misma y destacaremos las razones por las cuales las propuestas realizadas sirven para resolver algunos tipos de problema de mejor manera que otras posibles soluciones existentes. También analizaremos los posibles pasos a seguir y cuáles podrían ser algunas mejoras al presente trabajo en el futuro.

Apéndice A – Análisis Estadístico de los Datos

En este capítulo se afronta el problema de la generación de perfiles de usuarios desde el punto de vista estadístico y de aprendizaje de máquina. Analizamos con detenimiento el dominio al que circunscribimos el problema, presentando los atributos del dominio y sus relaciones, mediante una primera aproximación estadística.

Apéndice B – Framework

En este apéndice se muestra el diseño abstracto del framework utilizado para el análisis e implementación de las distintas implementaciones de nuestro agente.

Apéndice C – Referencias

En este último apéndice se encuentran las referencias utilizadas en esta tesis.

Capítulo II

2. Agentes

Normalmente en cualquier campaña de marketing o de comercialización masiva se tiene en cuenta el modelo llamado, por sus siglas, CBB (Consumer Buying Behavior). El modelo CBB describe el proceso de compra como una serie de seis estados fundamentales [KOH/2001], a saber: Identificación de la necesidad, Elección del producto, Elección del mercado, Negociación, Compra y entrega, Servicio del producto y evaluación. En estos entornos es posible utilizar la noción de perfil para describir las características particulares de un cliente (Ej, joven, masculino, estudiante). También es cierto que no sólo en estos entornos se utiliza la noción de Perfil de usuario, en cualquier tienda o supermercado puede verse distribuida la mercadería de manera tal que, en cada zona de distribución hay gente con similitudes de edad, sexo y posición económica, por ejemplo en la zona de plantas suele haber muchas más señoras mayores que hombres adolescentes.

Existen varias técnicas para definir perfiles a partir de los datos que genera el usuario durante la realización de la compra. Muchas de estas técnicas provienen del data mining y se utilizan históricamente para procesar información off-line y generar cluster de usuarios, donde cada cluster se identifica con un perfil, los cuales pueden identificarse con el mismo perfil. De manera similar existen algunas aplicaciones basadas en reglas las cuales aprenden y generan reglas del tipo

$$Si\ sexo=hombre \wedge edad=25 \rightarrow zapatillas\ deportivas == true.$$

Que indica si el sexo es hombre y la edad es de 25 le gustan las zapatillas deportivas. La mayoría de estos sistemas también trabajan de manera off-line. Con el surgimiento de la teoría de agentes, se genera un abanico de nuevas posibilidades. Como por

ejemplo utilizar agentes para automatizar algunos de los pasos del proceso de compraventa. En especial, mediante la generación de perfiles de usuarios que brindan información sobre los ítems del CBB. De esta manera tendríamos agentes aprendiendo perfiles de usuarios, utilizando técnicas complejas de Aprendizaje de Máquina y, exigiendo al agente la inteligencia necesaria para realizar esto. Como se puede apreciar luego del capítulo uno y esta breve introducción, en este tipo de solución ocurre una intersección entre distintas disciplinas, como ser Marketing, Agentes de Software, Aprendizaje de Máquina y Sicología.
En este capítulo se presenta una introducción al tema del trabajo realizado. En este capítulo se explicará: Qué es un perfil, qué es un agente y su historia, qué es aprendizaje de máquina, de qué trata el modelo CBB, la relación que une a todos ellos; y se presentará, además, una reseña del estado del arte en la generación y aprendizaje de perfiles de usuarios por parte de agentes.

2.1. Perfiles de Usuarios Para Conocer los Gustos de Nuestros Clientes

La noción de Perfil se utiliza para describir las características particulares de un cliente (Ejemplo, joven, masculino, estudiante). La idea de Perfil puede interpolarse a otras áreas, por ejemplo la sicología, en ella se definen perfiles para las personas, y los conocemos comúnmente como perfiles psicológicos, ejemplos son: agresivo, con complejo de inferioridad, anárquico, etcétera, y es común etiquetar a las personas con al menos una de estas categorías.
Existen varias técnicas para definir perfiles a partir de los datos que genera el usuario en donde realiza la transacción (en nuestro caso una aplicación de comercio electrónico) muchas de estas técnicas provienen del data mining y su utilización para procesar información off-line y generar clusters de usuarios los cuales pueden identificarse con un mismo perfil.
De manera formal, nos referimos a un Perfil de Usuario como a la representación de las necesidades del usuario, y podría implementarse como una estructura a través de la cual se definirá las características particulares de cada usuario, sus gustos, y preferencias.
Un perfil debería contener información suficiente para responder a preguntas tales como: ¿qué productos, está interesado en comprar el usuario?, ¿A qué precios?, ¿De qué calidad?, ¿En qué épocas del año? O ¿En qué talle y color?.

2.2. Técnicas de Construcción de Perfiles

Existen varias técnicas para la construcción de perfiles de usuario, las mismas, son propuestas surgidas recientemente. Entre ellas merecen mención las que pueden encontrarse en [RES/1997], [MAE/1995], [KON/1997], [BOR/1998] y [AGG/1999]
Dos paradigmas principales surgen a raíz de estos estudios y deben tenerse en cuenta a la hora de construir perfiles. Estos son: [LAW/2001]

2.2.1. Content Based

En este modelo se utiliza la similaridad de los ítems a seleccionar basados en los ítems seleccionados anteriormente. Típicamente ítems, y perfiles son computados de manera conjunta, utilizando alguna medida estándar de distancia entre ellos, como por ejemplo el Coseno.[LAW/2001]En aplicaciones de comercio electrónico se lo utiliza definiendo un espacio de atributos, mediante una taxonomía de los productos, unidos por un cierto grado de coerción; de esta manera, si un usuario compra un producto, es posible que quiera algún otro producto del mismo grupo. Para obtener el interés se mide el vector de atributos de importancia del usuario con el del grupo. El de menor distancia es el seleccionado.

Un problema usual de la aplicación de técnicas Content-Based es que producen sobre entrenamiento debido a que sesgan los productos hacia productos viejos obviando los productos más nuevos [LAW/2001]. Este problema, se presenta por las implementaciones actuales, basadas en algoritmos que necesitan grandes volúmenes de datos, y con una baja cantidad de productos nuevos, no dan resultados en relación al reducido tiempo en la tarea de procesamiento de repeticiones.

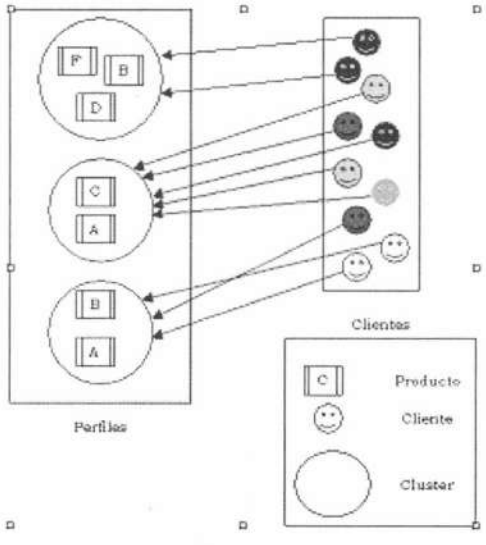

Figura 2.1 - Diagrama de un esquema de perfiles Content Based

En la *Figura* 2.1 se describe cómo se asocia un usuario (cliente) hacia un determinado perfil. La flecha indica aquel perfil que mejor representa al usuario, es decir, el de menor distancia.

2.2.2. Collaborative

En esta aproximación se utiliza la similitud entre perfiles, por ejemplo se recomienda un ítem a un usuario dado que un usuario con el mismo perfil compró el ítem. Esta técnica suele utilizarse también para definir estereotipos de perfiles en los que es asignado cada nuevo cliente (ídem al ejemplo de la sicología). De manera amplia podemos ver a esta técnica como la utilización de datos de un grupo o población de usuarios, para producir una recomendación individual. Ejemplos de estos son Ringo [MAE/1995]

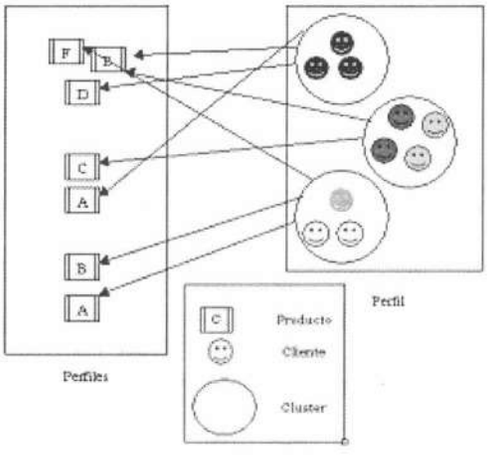

Figura 2.2 - Diagrama de un esquema de perfiles Collaborative

El diagrama muestra cómo se relacionan los grupos de usuarios con los productos.

2.2.3. Técnicas de Construcción de Perfiles Dinámicos

En la generación de nuestros Perfiles de Usuario, por tratarse del dominio del comercio electrónico, requerimos que la generación del Perfil se realice en la máquina cliente para respetar la privacidad de la información que observamos relativa a su comportamiento como cliente. Este perfil será personalizado y privado, es decir no tenemos clústeres, ni de usuarios, ni de productos, sino que cada usuario tendrá su propio dominio de interés, existiendo de esta manera, tantos Perfiles como clientes. Además, en este entorno, contamos con información observada, que no existe en otros clientes, por ejemplo, podemos conocer si un cliente eligió un producto en primera instancia y luego lo dejo por otro más barato.
Es decir, conocemos la historia completa de la compra o rechazo de un producto, sabemos que productos miró, las categorías de su preferencia, incluso sabemos que está mirando y haciendo en este momento, con lo cual pueden realizarse sugerencias pertinentes relativas a lo que el cliente está buscando.

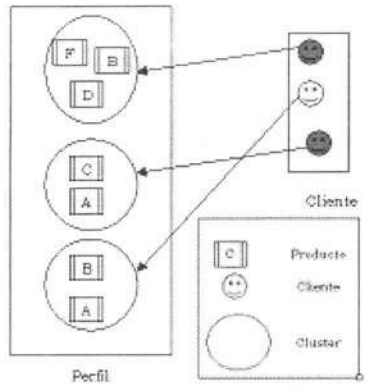

Figura 2.3 - Diagrama de Perfiles por usuario (Content Based)

El gráfico muestra cómo cada usuario se relaciona con un perfil específico.

Aun así podemos encajar esta técnica de generación de perfiles como content-based, debido a que el agente aprenderá de datos pasados, que se traducen a ítems seleccionados en el pasado. Los ítem pasados ayudan a definir las nuevas ofertas, con lo cual, se estaría utilizando cierta similaridad entre los ítems, y una agrupación de los ítems que exceden un determinado umbral en el valor de importancia inferido por el agente.

2.3. Aprender Perfiles Como un Problema de Aprendizaje de Máquina

Es válido enfocar la creación y mantenimiento de un perfil de usuario, como una tarea de aprendizaje. En este enfoque, la tarea es aprender el perfil que maximiza las posibilidades de compra de un cliente.
En esta sección se explica qué es aprendizaje de máquina y cómo lo utilizamos para aprender un perfil.

2.3.1. Aprendizaje de Máquina

El Aprendizaje de Máquina (Machine Learning) es la automatización de un proceso de aprendizaje. El aprendizaje de máquina como campo de investigación agrupa conceptos e ideas de varias ramas de la ciencia, como por ejemplo, sicología, estadística, neurobiología, teoría de la información, data mining, base de datos, KDD y muchos otras más.

2.3.2. Definición de Aprendizaje

> Diremos que un programa aprende de una Experiencia E, con respecto a alguna clase de tarea T y una medida de performance P, si su performance en la tarea T medida por P, mejora con la experiencia E.

Como un ejemplo de aprendizaje, considere un programa que aprende a jugar a las damas. Éste mejora su performance como medida por su habilidad para ganar en las clases de tareas compuestas por partidas de damas jugadas, a través de la experiencia obtenida jugando partidas consigo mismo.
O sea en este ejemplo podemos identificar Tarea, Medida y Experiencia como.

> Tarea T: Jugar Damas
> Medida de performance P: Porcentaje de partidas ganadas al oponente.
> Experiencia de entrenamiento E: Realizar juegos consigo misma.

De esta manera, podemos clasificar nuestro problema particular de aprender perfiles dinámicos de comercio electrónico como:

> Tarea T: Aprender Perfiles para sugerir productos a los usuarios del sistema.
> Medida de performance P: Cantidad de sugerencias aceptadas.
> Experiencia de entrenamiento E: Observación del uso del sistema por parte del usuario. Sugerencias aceptadas, sugerencias rechazadas y sugerencias ignoradas.

2.3.3. Elección de Técnicas de Aprendizaje

Las técnicas seleccionadas para esta implementación son tres, a saber: redes bayesianas, algoritmos genéticos, y redes neuronales. Estas técnicas fueron seleccionadas dado que, en las experiencias reportadas, se presentan ventajas en relación con la obtención de perfiles con mayores detalles de relaciones entre datos, necesarios para que agentes de interfaz puedan actuar más eficientemente.
Dos grupos de métodos no son considerados en nuestro análisis. Un grupo es el formado por métodos que permiten formar clusters, y en general, construir reglas simples a partir de éstos. Este grupo no fue considerado ya que sólo es aplicado para agentes con una funcionalidad trivial y ya suficientemente explorada. El segundo grupo que no es considerado es aquel que está formado por algoritmos de análisis de datos que requieren grandes volúmenes de información para ser efectivos, algo que no ocurre en el caso de agentes de interfaz que tienen que comenzar a construir perfiles desde el comienzo del uso del sistema, contando obviamente con muy pocos datos.
En relación con las tres técnicas que se analizarán, otra característica importante es el carácter evolutivo de las técnicas, es decir que puede ir mejorando iterativamente.

2.3.3.1. Técnicas de Computación Evolutiva

La computación evolutiva se basa en el empleo de modelos de procesos evolutivos para el diseño e implementación de sistemas de resolución de problemas. Los distintos modelos computacionales que se han propuesto dentro de esta filosofía suelen recibir el nombre genérico de Algoritmos Evolutivos.
Existen cuatro tipos de algoritmos evolutivos bien definidos que han servido como base a la mayoría de los trabajos realizados en el área: los algoritmos genéticos, las estrategias de evolución, la programación evolutiva, y la programación genética. Aunque desde un punto de vista más amplio, también pueden considerarse como técnicas evolutivas las redes neuronales, las redes bayesianas adaptativas, y aprendizaje por refuerzo, dadas sus características iterativas y adaptativas.

Por lo tanto, la selección de técnicas, se basa, en aquéllas cuyo poder de representación, aprendizaje y adaptación ya haya sido demostrado. Situación que ocurre con las tres técnicas elegidas, las cuales son de las más viejas y las más interesantes y sobre todas las cosas muy testeadas. Sabemos que son robustas, capaces de representar dominios de muchas variables y todas ellas pueden partir desde la falta total de conocimiento e ir aprendiendo paulatinamente.

La elección de las técnicas de aprendizaje a utilizar será justificada debidamente más adelante, en el Capítulo III.

2.4. Agentes de Software para Generar Perfiles Dinámicos de Usuarios

Tenemos que la gestación de perfiles de usuario encuentra su cuna en el comercio electrónico, dadas las características particulares de éste. Así que nos encontramos ante el problema de crear perfiles que sean dinámicos y adaptables de manera efectiva. Debemos tener en cuenta que, como mencionamos anteriormente, debemos aprender perfiles mediante la observación. En este contexto, los agentes de software son efectivos para llevar a cabo esta tarea.
El concepto de agentes de software se describe a continuación.

2.4.1. Agentes de Software

Existen muchas definiciones de Agentes de Software, algunas de ellas consistentes, y otras no. Lo cierto, es que el término agente se propagó a lo largo de distintos ambientes, tanto comerciales como académicos, con lo cual se creo un basto y heterogéneo campo de investigación y desarrollo, en los que se utiliza la palabra agente de manera igualmente heterogénea.
Por esto se hace necesario apegarse a una definición, creemos que la más consistente es la de P. Maes, una de las pioneras en lo que a investigación en el campo de los agentes respecta. Ella definió un agente autónomo como:

> Un sistema computacional que habita en un entorno dinámico y complejo, censa y actúa autónomamente en ese entorno, alcanzando un conjunto de objetivos o tareas para los cuales fue diseñado [MAES/1995].

Aunque existen otras definiciones, merecen mención por su aceptación en la comunidad las que pueden encontrarse en: [RUS/1995], [SMI/1994], [WOO/1995], [FRA/1996], [SHO/1997].
Con esta definición establecida, podemos entrar en la historia de los agentes, en la siguiente sección.

2.4.2. Historia de los Agentes

Desde el inicio de la historia registrada, las personas han estado fascinadas con la idea de tener asistentes no humanos. Nociones populares de androides, humanoides, robots, cyborgs, y otras criaturas de ciencia-ficción han cultivado inconscientemente la forma en que los agentes de software son percibidos. Todos los usuarios de computadoras han tenido el sueño de un "mayordomo digital". Sin embargo, las

experiencias diarias de los usuarios de computadoras con los misterios del software ordinarios, llenos de errores molestos, funcionalidades incomprensibles, y virus peligrosos, refuerzan la idea de que si el software tiene el poder de autonomía e inteligencia, traerían más problemas que soluciones. Estos problemas latentes son los que deben ser tenidos en cuenta cuándo se diseñan los agentes de software[NWA/1996].
Para encontrar los principios de la investigación en agentes, podemos remontarnos a los años 70 en el marco de la investigación en la Inteligencia Artificial Distribuida (DAI). Carl Hewitt propuso un modelo en el cual se notaba con el nombre de Actor a un objeto, el cual era autocontenido, interactivo, de ejecución concurrente, con un estado interno encapsulado, y el cual podía recibir y enviar mensajes de manera similar a un objeto [NWA/1996], concepto similar al de agente en sus características más básicas.
Se puede dividir el desarrollo de la teoría de agentes en dos períodos: el primero, en la década del 80', y el segundo, en la década del 90'. El primer período surge de la "Inteligencia Artificial Distribuida", se centró principalmente en agentes del tipo deliberativo con modelos simbólicos internos, y contribuyó a comprender problemas tales como interacción y comunicación entre agentes, la descomposición y distribución de las tareas, coordinación y cooperación, y resolución de conflictos vía negociación. El segundo período, en cambio, es un movimiento que crece rápidamente para estudiar un rango mucho mayor de agentes, desde el más simple hasta los moderadamente inteligentes. Se desarrollaron teorías de agentes, arquitecturas y lenguajes, con una diversidad de aplicaciones y enfoques, concentrándose en la función, y acción remota.
La tecnología de agentes integra diversas disciplinas de las Ciencias de la Computación, tales como orientación a objetos, arquitecturas distribuidas, sistemas de aprendizaje adaptativo, inteligencia artificial en general, sistemas expertos, algoritmos genéticos, procesamiento distribuido, algoritmos distribuidos, entornos sociales en línea, seguridad, y varios más. Esta integración ha hecho de los agentes un interesante tópico de estudio [NWA/1996].
Las mayores aplicaciones de esta tecnología se dan en Internet, un sistema abierto que se caracteriza por los cambios dinámicos de las estructuras del sistema, compuesta por partes desconocidas de entrada, heterogéneas y cambiantes, que necesariamente necesitan interactuar. El crecimiento de Internet y la cantidad cada vez mayor de usuarios promovieron el desarrollo acelerado de los agentes.
Actualmente, los agentes se aplican en inteligencia artificial distribuida, robótica, vida artificial, computación distribuida orientada a objetos, interacción hombre - máquina, interfaces inteligentes y adaptativas, búsqueda y filtrado inteligente, obtención de información, adquisición de conocimientos, programación de usuario final, programación por demostración, simulaciones científicas, soporte para comunidades virtuales, soporte para empresas virtuales, soporte para toma de decisiones, administración de work-flows, software educativo, manejo de transacciones en el comercio electrónico, administración de redes, control de tráfico aéreo, data mining, filtrado de correo electrónico, bases de datos inteligentes, y muchas áreas más.

2.4.3. Características Principales de los Agentes

Las características principales son:

Cada agente está situado o es parte de un entorno. Cada uno censa ese entorno y actúa en forma autónoma sobre él. Cada uno actúa según su propio cronograma, satisfaciendo los objetivos para los cuales fue diseñado, y/o evolucionado como los

humanos y animales. Cada uno actúa tal que su acción puede afectar lo que más tarde censa, es decir, su acción afecta el entorno. Cada agente se mantiene "vivo" hasta que él mismo decide no hacerlo; no siempre el humano lo puede matar, como en el caso de los agentes móviles.
Un resumen de las características más importantes son las siguientes:

- Autonomía: habilidad de actuar y operar por sí solo, en forma periódica, espontánea, por iniciativa propia, y sin intervención y guía de un humano.
- Orientación a objetivos / Proactivo: habilidad de "tomar la iniciativa" y actuar con un propósito, no simplemente responder al entorno. Es decir, no necesita de mensaje para realizar una acción sino que puede realizarla por propia voluntad.
- Reactivo: habilidad para censar y medir selectivamente los cambios en el entorno, y actuar según sus objetivos.
- Persistencia: continuidad temporal de la identidad y el estado durante largos periodos de tiempo.
- Comunicación: habilidad de comunicarse con personas y otros agentes.
- Colaboración / cooperación: capacidad de comunicarse con otros agentes para cumplir una tarea en conjunto.
- Movilidad: habilidad para moverse en forma auto-dirigida desde una plataforma de host a otra.
- Aprendizaje: capacidad de aprendizaje a medida que reaccionan e interactúan con el entorno externo, tal que con el tiempo y la experiencia, mejore su performance y comportamiento.
- Benevolencia: su comportamiento ayuda al usuario (no es un virus), y es confiable, no miente.
- Flexibilidad / Inferencia: capacidad de actuar sobre especificaciones abstractas de tareas, utilizando conocimiento previo sobre objetivos generales y métodos preferenciales para lograr flexibilidad; ir más allá de la información dada, y puede tener modelos explícitos de sí mismo, del usuario, de la situación, o de otros agentes.
- Deliberación: contar con un modelo de razonamiento interno, para planear y negociar con otros agentes para lograr sus objetivos.
- Personalidad: aptitudes mentales tales como creencias, deseos, intenciones, y actitudes emocionales.

2.4.4. Clasificación de Agentes

Dada las características enumeradas y el amplio campo de investigación existente se puede observar que existe una gran variedad de diferentes tipos de agentes. Por esto así como existen múltiples definiciones de agentes existen también múltiples taxonómias, entre ellas podemos mencionar: [FRA/1996], [GIL/1995], aunque aquí presentemos la siguiente:
En [NWA/1996] se define una clasificación a partir de tres atributos: autonomía, aprendizaje, cooperación. A partir de estas características, deriva diferentes tipos de agentes: colaborativos, colaborativos que aprenden, de interface, e inteligentes. (*Figura 2.4*)

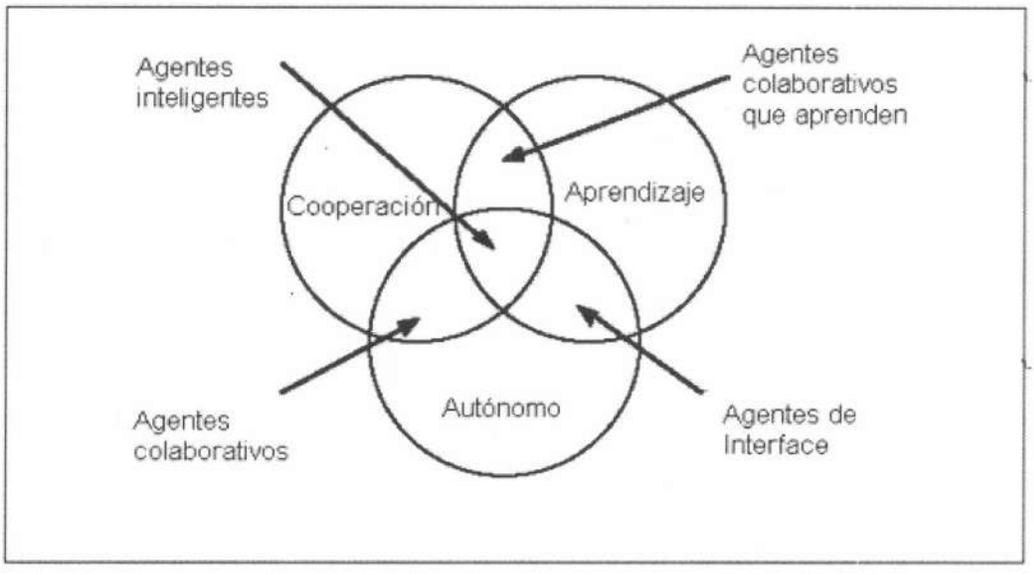

Figura 2.4 - Topología de Agentes

Toda la taxonomía nace de estos atributos y de las conjunciones de estos. La cual describimos a continuación con detalle:

1. *Agentes Inteligentes:*

Un agente inteligente debe cumplir con las siguientes características: cooperación, aprendizaje, y autonomía. Estas tres características deben ser iguales en importancia y complejidad para poder estar en presencia de un agente inteligente.

2. *Colaborativos:*

Este tipo de agente hace principal hincapié en la autonomía y cooperación del agente con otros agentes. Pueden aprender pero este aspecto no suele ser de mucha importancia. Ellos pueden negociar entre sí para lograr sus objetivos.

Las categorías más importantes son: autonomía, habilidad social, sensibles al contexto y proactivos.

3. *Colaborativos que aprenden:*

Este tipo de agentes también se basa en la colaboración, pero, en vez de poner énfasis en la autonomía lo pone en el aprendizaje. Lo que tampoco quiere decir que no tengan algo de autónomos, simplemente no hacen hincapié en ello.

4. *Agentes de Interfaz.*

Este tipo de agente se centra principalmente en la autonomía y el aprendizaje, no importando demasiado la cooperación y comunicación con otros agentes. La idea es la de un asistente personal que colabora con el usuario. Cabe destacar la diferencia entre cooperación con el usuario y cooperación con otros agentes, en el primer caso puede no requerirse un lenguaje de comunicación explícito. Simplemente el agente

entiende las acciones del usuario y lo puede realizar principalmente de cuatro maneras [MAE/1994]:

- Mediante la observación e imitación del usuario.
- A través de recibir feedbacks de los usuarios, ya sea, positivos o negativos.
- Mediante instrucciones explícitas del usuario.
- Mediante el consejo de otro/s agente/s.

De esta manera, un Agente de Interfaz no se limita solamente a responder preguntas del usuario sino que presenta una actitud proactiva actuando y aprendiendo por su cuenta. Pudiendo, en ciertas condiciones, programarse el mismo a lo largo del tiempo [MAE/1994], es decir, evolucionando. Con estos atributos los Agentes de Interfaz tenemos que éstos además están dotados de capacidad de decisión, capacidad de aprender, capacidad de entender los pedidos del usuario, y capacidad de actuar de manera independiente ante estos pedidos / eventos. Características idóneas para abordar el problema planteado.

De manera ortogonal a esta clasificación también se presenta la siguiente:

1 *Agentes móviles*

Se definen a los agentes como procesos de software que recorren la red, mudando de tiempo en tiempo de un host a otro. Estos pueden ir de un host a otro para luego regresar a casa simplemente o pueden vagar constantemente y enviar la información recogida a algún lugar. Un ejemplo de este tipo podría ser el de un agente de reservación de vuelos. Aunque si sólo se destaca el atributo movilidad, esta única característica no lo convertiría en un agente, sino que debe tener, además, atributos como cooperación y/o autonomía.

2 *Agentes de información / Internet.*

Este agente surge de la necesidad de manejar la gran cantidad de información, que existe en una, cada vez mas, desarrollada Internet. La meta de éstos es manejar, manipular y seleccionar información proveniente de muchas fuentes distribuidas.

3 *Agentes Reactivos*

Los agentes reactivos se caracterizan principalmente por el hecho de no poseer un modelo simbólico interno de su ambiente, sino que actúa o responde según un modelo estimulo-respuesta para presentar el estado en que se encuentra embebido. Se caracterizan por ser relativamente sencillos e interactuar con otros agentes, también, de manera sencilla.

2.5. Agentes Interfaz Para Aprender Perfiles de Usuarios

Podemos utilizar agentes interfaz para aprender perfiles, como dijimos anteriormente queremos aprender de manera que el agente se transforme en un asistente personal del usuario y que aprenda observando sobre el hombro del usuario [MAE/1994]. Para esto, el agente debe tener en cuenta cuál es el comportamiento de un cliente en una transacción. Siempre, durante una transacción, se ejecutan una serie de pasos,

independientemente del tipo de la aplicación; y deberá utilizar esta información durante la observación. Para esto utilizamos el modelo CBB que se describe más abajo.

2.5.1. Comportamiento de Compra

El modelo CBB (Consumer Buying Behavior, Comportamiento de compra del consumidor) describe el proceso de compra como una serie de seis estados fundamentales: [KOH/2001]

Identificación de la necesidad: Caracteriza al comprador enterándose de alguna necesidad no conocida. El comprador puede ser motivado a través de información acerca del producto.
Elección del producto: Incluye la obtención de información para determinar qué comprar. Incluye la evaluación de alternativas de productos basada en el criterio del comprador. El resultado es el "conjunto de consideración" de los productos.
Elección del Mercado: Combina el "conjunto de consideración" del punto anterior con información especifica de mercado para determinar quién compra en qué lugar. Esta etapa también incluye la evaluación de alternativas de mercado, basadas en los criterios provistos por el comprador (tales como precio, garantía, disponibilidad, tiempo de entrega y reputación).
Negociación: Considera los acuerdos en términos de una transacción. La negociación varía en duración y complejidad dependiendo del mercado. En los mercados tradicionales de reventa, los precios y otros aspectos de la transacción están a menudo prefijados, no dejando lugar a negociación.
Compra y entrega: Señala la terminación de la etapa de negociación. En algunos casos, la forma de pago u opciones de entrega puede influenciar el producto y/o corretaje del mercado.
Servicio del producto y evaluación: Involucra el servicio posventa, servicio al cliente y evaluación de la satisfacción de toda la experiencia de la compra.

2.5.2. Como se Relacionan los Agentes con el CBB

La tecnología de agentes está siendo utilizada para automatizar algunos de los pasos del proceso de compraventa. Los agentes pueden actuar como mediadores en el proceso de compraventa. Sus características de autonomía, persistencia y personalizabilidad los hace apropiados para mediar al consumidor en los procesos de búsqueda y filtrado de información, evaluación personalizada, coordinación compleja de la transacción, interacciones basadas en tiempo, etcétera, que se corresponden con los cuatro primeros pasos del modelo de Compra que presentamos.

2.5.3. Algunas Aplicaciones de Agentes y su Utilización.

Para entender todo lo expuesto en esta sección, lo mejor es dar algunos ejemplos. A continuación se presentan una serie de implementaciones de agentes junto con la categoría, según la taxonomía dada, a la que pertenecen. En algunos de ellos puede verse cómo actúan automatizando algunos de los pasos de CBB.

2.5.3.1. NewsAgent [AMA/ 1999]

NewsAgent es un tipo de agente inteligente que tiene la capacidad de generar noticias personalizadas, a partir de las preferencias particulares de un usuario. Los datos sobre los gustos son extraídos mediante la observación del usuario. Este agente utiliza una clasificación estática del mundo y Razonamiento Basado en Casos (CBR) para la subclasificación dinámica de los documentos interesantes.

2.5.3.2. PersonalSearcher [GOD/ 2000]

PersonalSearcher es un agente personal que asiste al usuario en la búsqueda de documentos interesantes en la WWW. Este agente lleva a cabo búsquedas paralelas en las más populares máquinas de búsqueda de la web y filtra su resultado, listando al usuario un reducido número de documentos con la más alta probabilidad de ser relevantes al usuario. Este filtro está basado en un perfil de usuario que el agente construye mediante la observación de las acciones del usuario en la WEB. El agente utiliza razonamiento basado en caso para detectar que subconjuntos específicos son de interés al usuario y los ordena en una jerarquía que conforma el perfil.

2.5.3.3. Obelix

Obelix es un planificador inteligente de reuniones que automáticamente organiza las actividades del usuario. Tomando en cuenta sus preferencias, los conflictos entre grupos, Distancias de viaje, etcétera.
Para esto utiliza Razonamiento Basado en Casos aprendiendo a través de experiencias pasadas del usuario.

2.5.3.4. ROLAP

ROLAPAgent es un agente inteligente el cual recibe como input una consulta (query) multidimensional. Entonces la transforma en un conjunto de consultas SQL optimizando la performance. Estas optimizaciones son basadas en heurísticas y en experiencias pasas aprendidas por el agente.
ROLAPAgent Utiliza Case Base Reasoning para aprender de experiencias pasadas. Estos casos son utilizados para producir la más eficiente consulta SQL equivalente a la consulta multidimensional recibida como input.

2.5.3.5. Robotic Soccer: Fútbol Simulado

En [KIT/ 1997] se presenta un sistema multiagente llamado Robotic Soccer, y aunque se trate de un juego, mantiene la mayoría de las complejidades del mundo real. Las razones para utilizar un sistema de multiagente, se debe a que se requiere que cada uno de los jugadores sea representado como una entidad en sí misma, que tiene capacidad de decisión, que afecta el comportamiento de otros y que tienen una visión limitada del escenario del juego.
En este contexto, los jugadores pueden ser reactivos o deliberativos en algún grado. Los agentes que los representan si son extremadamente reactivos pueden simplemente mirar la pelota y moverse hacia ella y patear cuando sea posible, no siendo necesario tener conocimiento alguno de que son parte de un equipo. Por otro lado los jugadores

pueden moderarse uno a otros, o sea activando un razonamiento deliberativo sobre si acercarse a la pelota o si moverse a una parte distinta del campo de juego para defender o recibir un pase logrando que los jugadores razonen sobre cómo se afecta el comportamiento entre ellos en un ambiente dinámico.

2.5.3.6. YAMS: Sistema de Manufacturas

En [VAN/1987], se describe un sistema multiagente llamado YAMS (Yet Another Manufacturing System). La motivación de su desarrollo se debe a que una empresa de manufacturas se compone de celdas de trabajo autónomas que son la base de una estructura jerárquica con un proceso de producción con parámetros que cambian constantemente, como ser los productos a ser manufacturados, los recursos disponibles, las restricciones de tiempos, etcétera.
El objetivo principal de YAMS es el de manejar el proceso de producción de las plantas de manufacturas y para ellos se adopta un modelo de sistemas multiagentes basados en el protocolo Net Contract.
YAMS asigna un agente para cada una de las celdas de trabajo y para cada una de las entidades perteneciente a la jerarquía antes mencionada (como ser grupos de celdas de trabajo, grupos de manufacturas, grupos de fabricas, etcétera)., Cada uno de estos agentes tiene un conjunto de planes que representan sus capacidades y a través del protocolo Contract Net se pude delegar tareas a las entidades dependientes.

2.5.3.7. Guardián: Monitoreo de Pacientes

En [HAY/1989], se describe un sistema multiagente llamado GUARDIÁN. El sistema fue motivado por dos razones: Primero, que el modelo de cuidado de pacientes en una Unidad Quirúrgica de Cuidados Intensivos (UQCI) es esencialmente el de un equipo, que reúne a especialistas de diversas áreas que cooperan para organizar el cuidado de la salud de los pacientes. Segundo, que uno de los factores importantes en el cuidado de pacientes en una UQCI, es compartir en forma adecuada la información entre los miembros del equipo.
En particular los especialistas, no tienen la posibilidad de monitorear minuto a minuto el estado del paciente, relegando esto a las enfermeras, quienes con frecuencia interpretan incorrectamente esta información por falta de la experticia, que el especialista tiene.
El sistema GUARDIÁN, utiliza multi agentes organizados por jerarquías, y los tipos de agentes que aparecen son: los agentes de percepción / acción, los agentes de decisión y el agente de control.

2.5.3.8. Warren: Manejo de Portafolio Financiero

En la práctica, el portafolio financiero es llevado a cabo por las casas de inversiones que emplean a grupos de especialistas para recolectar, filtrar y evaluar información relevante. Luego basándose en sus evaluaciones y en predicciones de la futura economía, hacen sugerencias sobre la compra o venta de bonos, fondos, stocks, etcétera.
Los multiagentes se adaptan a esta situación dada la variedad y la cantidad de fuentes de información y las diferentes experticias que deben ser tenidas en cuenta para realizar una buena recomendación (por ejemplo, comprar o vender).

El trabajo de manejo de portafolios, se descompone en varias tareas, siendo estas, el aprendizaje o selección de la información del perfil de usuario, la recolección de información sobre la situación del portafolio del usuario, y sugerir y monitorear una reubicación para juntar el perfil actual del usuario y los objetivos de inversión.
Warren es un sistema que surge de la aplicación de un framework llamado RETSINA y los agentes que lo componen son: Portfolio Manager Agent (interactúa con el usuario y toma la información de su perfil y objetivos), Fundamental Análisis Agent (interpreta el stock desde un punto de vista determinado), Technical Analysis Agent (usa técnicas numéricas para predecir el futuro cercano del mercado), Breaking News Agent (filtra noticias y selecciona aquéllas que son más importantes según el perfil del usuario y que afecten al manejo del portfolio), Analyst Tracking Agent (trata de reunir la inteligencia sobre lo que piensan los analistas a cerca de una compañía), Stock Tracker Agent (monitorea las fuentes de reportes de stock).

2.6. Comentarios Finales

Sabemos qué es un agente, y cuáles son las aplicaciones más difundidas de estos, conocemos además la complejidad de generar y mantener perfiles de usuario. Precisamente es esta complejidad la causante de que métodos sencillos fracasen o logren medidas de éxito relativamente pobres.
Pero el tema de la complejidad es uno sobre el que todavía falta aclarar varios puntos. El principal es determinar el tipo de dominio para entender la complejidad de éste.
Luego un análisis previo de los datos históricos de la aplicación podría traer cierta luz a los efectos de entender esta complejidad. Por último necesitamos los fundamentos teóricos necesarios para entender las propiedades de los distintos algoritmos y la caracterización frente al problema a resolver. Esto es lo que veremos en el capítulo siguiente.

Capítulo III

3. Vendor un Agente Para Comercio Electrónico

En este capítulo presentamos a *Vendor*, detallaremos su diseño, y las tareas que realiza. *Vendor* es un agente de interfaz para comercio electrónico, capaz de generar perfiles de usuarios, utilizando distintas técnicas de aprendizaje. Este agente será utilizado para testear 3 algoritmos de aprendizaje de máquina y su desempeño en entornos de comercio electrónico.
Vendor realiza varias tareas, entre ellas, construir y mantener un perfil de usuario, que representa los gustos y hábitos particulares de su usuario. Sugerir, mediante la utilización de este perfil, sólo aquellos productos que son del agrado del usuario.
Para lograr este objetivo hay varios puntos a tener en cuenta, como por ejemplo, cuál es la información qué puede tomar el agente, cómo la toma, cómo la interpreta; de qué manera realiza las sugerencias, cómo podemos hacer qué la implementación del agente sea independiente de la técnica de aprendizaje utilizada, y finalmente, cuál es el hábitat del agente, y cómo se desempeña en él. Estos temas son los que desarrollaremos con detenimiento en este capítulo.

3.1. Vendor

Como dijimos anteriormente, en el capítulo II, agentes de interfaz cumplen una importante función asistiendo al usuario en tareas repetitivas; y ayudándolo a manejar sobrecarga de información. Para el desarrollo de estas tareas, el agente debe adquirir el conocimiento necesario. Normalmente se capacita al agente con un conocimiento

básico para que pueda comenzar a desarrollar sus tareas, y luego, deberá aprender gustos y preferencias particulares de su usuario. Para poder aplicar este tipo de aprendizaje deben satisfacerse dos condiciones fundamentales[MAR/1997]. Primero, el uso del sistema debe envolver una considerable cantidad de tareas repetitivas. Segundo, estas tareas repetitivas son potencialmente distintas para cada usuario. Esta aproximación está basada en la idea de un Asistente Personal. Asistentes personales realizan, gradualmente, con mayor eficiencia, sus tareas. Mostrando así, el aprendizaje de los hábitos, intereses y gustos del usuario; y la adaptación del agente a estos con el tiempo.
Históricamente, para implementar el aprendizaje en este tipo de agentes, se utilizaron técnicas sencillas, o ingenuas. Técnicas que resultan insuficientes para entornos donde la cantidad de variables en juego es muy grande, la complejidad de la tarea a aprender es alta, y dónde no hay respuesta directa del usuario hacia el agente (feedback). Este es uno de los motivos por los cuales proponemos realizar una análisis de técnicas de aprendizaje complejas en dominios de comercio electrónico.
Vendor es un agente que aprende perfiles de usuarios dinámicos para comercio electrónico, utilizando distintas técnicas de aprendizaje de máquina. *Vendor* opera como un secretario personal del usuario.
Cada usuario tiene un agente que lo asiste. *Vendor* tiene la inteligencia suficiente para aprender a través de la observación del usuario del sistema. Automáticamente crea y administra un perfil para el usuario; en ese perfil se encuentra toda la información necesaria sobre el usuario, y que es de utilidad a la tarea que realiza el agente. *Vendor* aprende, únicamente, de la observación de las acciones del usuario, al momento de recorrer el sitio WEB de comercio electrónico.
El agente asiste al usuario ofertando aquellos productos que, según su criterio, le serán de mayor interés al usuario. El agente, también, interactúa con la aplicación pidiendo productos y ofertas para clasificar, según su importancia; y que serán ofrecidas al usuario en el momento que lo considere necesario; es decir, por propia voluntad, sin la intervención del usuario o de la aplicación.
La implementación de este agente permite aprender mediante distintas técnicas de aprendizaje. Es decir la técnica es configurable, esto permitirá en un futuro poder testear otras técnicas complejas de aprendizaje de máquina.

3.1.1. Vendor - Capacidades

Para poder analizar las técnicas de aprendizaje debemos en primera instancia, referirnos a algún dominio de comercio electrónico en particular. Seleccionamos para nuestro agente el de una tienda virtual de deportes. En ésta se realiza la venta de artículos deportivos a través de Internet. La elección del dominio se debe a que contamos con datos reales, provenientes de la empresa Muni Deportes S.A. Estos datos nos permitirán analizar la relevancia y magnitud de los atributos involucrados.
Vendor realiza principalmente cuatro tareas. La primera, es la construcción de un perfil del usuario al que asiste, esta tarea involucra varias subtareas de mucha importancia, como por ejemplo, observación del usuario, clasificación de las acciones del usuario, representación de intereses, y del entorno en que habita. La segunda tarea, es brindar al usuario aquellos productos que le son de interés, estas ofertas se harán a voluntad del agente, y no es necesaria la intervención del usuario ni de la aplicación para que ésta se produzca. Tercero debe ser capaz de aprender utilizando distintas técnicas de aprendizaje. Cuarta. Deberá mantener también un registro histórico del éxito obtenido,

según la técnica de aprendizaje utilizada. En las siguientes secciones explicáremos en detalle cada una de estas tareas.

3.2. Construcción del Perfil de Usuario

Como ya dijimos, un perfil de usuario es la representación de las preferencias y hábitos del usuario. En nuestro caso está formado por las variables involucradas en la aplicación de comercio electrónico a utilizar, más la fecha y la clasificación presunta del usuario (hombre, mujer, etcétera)

3.2.1. Observación

EL agente observa todas las operaciones realizadas por el usuario, mantiene y actualiza información relativa a los productos que compra, a los productos que agrega al carrito, o selecciona para una vista detallada; así como también, las ofertas rechazadas, o aceptadas. Además, para cada transacción realizada, se tiene en cuenta la fecha y hora de esta; para poder indagar factores como, temporalidad y, rutinas de compra en el usuario.
Cada una de las acciones realizadas por el usuario involucra una cierta cantidad de información que debe ser tenida en cuenta por el agente a la hora del aprendizaje.
A esta información la llamamos dominio y la describimos de manera formal a continuación.

3.2.1.1. Dominio

Nos referimos al dominio, como a la caracterización de todas las variables que involucra la tarea a aprender. En nuestro caso, trataremos con una función que va de una cierta cantidad de atributos a algún valor real representando la importancia del producto.
De manera Formal:
Sea:

$$D = \{ \langle x, f(x) \rangle \ / \ x \in V_1 \times V_2 \times \ldots \times V_n \} \qquad (3.1)$$

el conjunto de instancias de entrenamiento, donde:

V_i es el conjunto de valores posibles del i-ésimo atributo de las instancias

$$f: V_1 \times V_2 \times \ldots \times V_n \rightarrow C \qquad (3.2)$$

es la función de clasificación

C es el conjunto de clasificaciones posibles de las instancias.
Deseamos encontrar una hipótesis h que.

$$(\forall \langle x_i f(x_i) \rangle \in D) \ (h \wedge x_i) \ |\!- f(x_i) \qquad (3.3)$$

En nuestro caso, encontramos que la *f* a resolver es la siguiente. Dada una cantidad de atributos referentes a artículos deportivos y fechas. Tratar de predecir el grado de interés despertado en el usuario.
Por lo tanto *C* es real y la podemos definir en el intervalo [0,1]. Con respecto al Dominio de nuestra función. Podemos ver en la *Tabla 3.1* un breve resumen de cada uno de ellos, sobre un total de 16 atributos.

Nombre	Descripción	tipo dominio
Categorías	Categoría del producto	Discreto
Accesorios	Categoría accesorios	Discreto
Calzado	Categoría calzados	Discreto
Deportes	categoría deportes	Discreto
Entrenamiento	categoría entrenamiento	Discreto
Indumentaria	categoría indumentaria	Discreto
Marcas	categoría marcas	Discreto
CategoríaComprador	a quién está destinado el producto	Discreto
Precio	Precio del producto	Continuo
Catprecio	categoría del precio	Discreto
Exposición	tiempo de exposición	Continuo
Color	Colores	Discreto
RankingInterno	Categorización del desempeño del producto dentro de la empresa	Discreto
CuandoCompro	tiempo transcurrido desde la compra	Continuo
Calidad	calidad del producto	Discreto
Talle	talle del producto	Discreto

Tabla 3.1 - Atributos Involucrados en el dominio de la aplicación

Además tendremos en cuenta el atributo Fecha, que corresponde a la fecha y hora en que se realizó la transacción; y el atributo *Veces,* correspondiente a la cantidad de veces que realizó la consulta. Esto da una idea de relevancia de lo que el usuario está buscando.

Como podemos ver, 4 de los atributos son continuos, mientras que los otros 14 son discretos, también podemos ver que el rango de los atributos discretos es, en general, pequeño; esto podría ayudar en la tarea de análisis previo y estadístico, ya que, de existir correlaciones, las encontraríamos más fácilmente que si estos fueran numerosos.

3.2.2. Casos

Toda la información recogida por el agente en una acción del usuario es transformada en un caso, el cual será aprendido por el agente. Para esto el agente utiliza la metáfora de "Mirar sobre el hombro del usuario" [MAE/1994]. Se pone en lugar de éste y analiza los datos desde su perspectiva.
Para tener una idea clara de esto podemos ver en la *Figura 3.1* una imagen de la barra de navegación de la aplicación(resaltado con un circulo verde). En ella se puede ver las

variables a tener en cuenta por el agente, si un usuario realizará una búsqueda por la *Categoría =Indumentaria,* y *SubCategoría =Camisetas*, se le dará al agente una nueva lección o caso destacando el agrado por estas categorías. En la *Figura 3.2.* se muestra como queda armado el caso. Estos casos serán utilizados por el agente para aprender el perfil del usuario al que asiste.

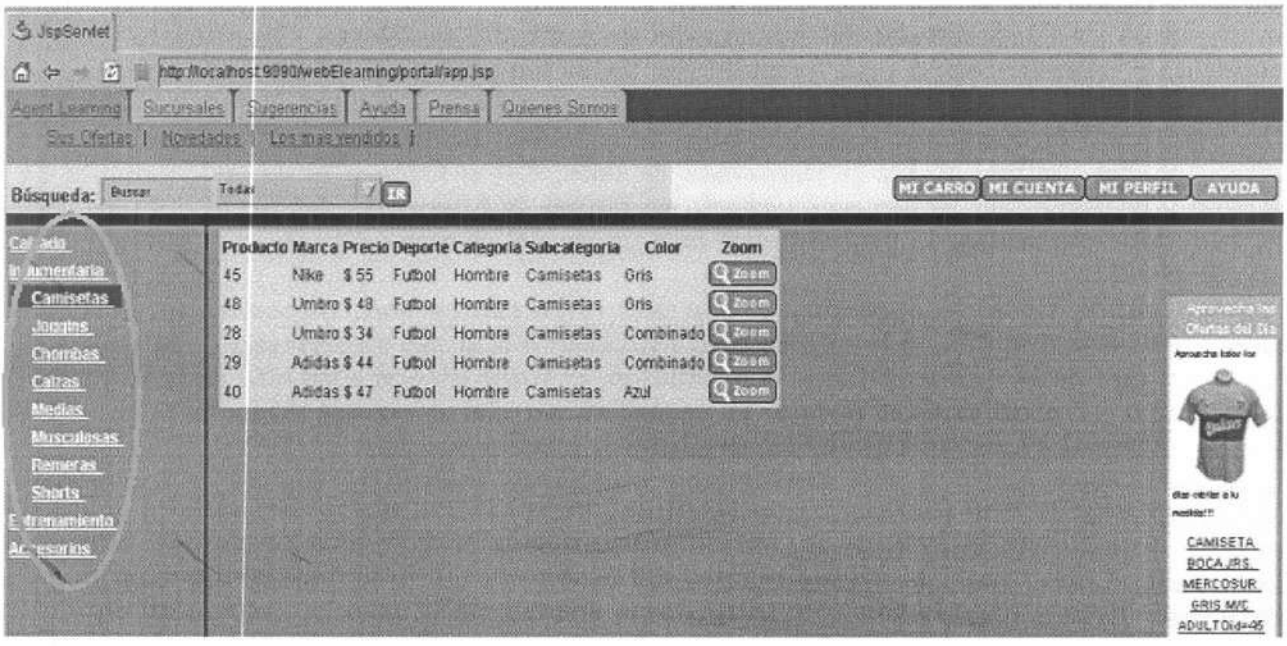

Figura 3.1 - Vista de la Barra de navegación de la aplicación

Usuario: PepeTrueno

Variables
Categoría: Indumentaria
Subcategoría: Camisetas

Fecha : 12/12/2202
Hora: 23:45

 # de Veces que realizó esta consulta: 1

Figura 3.2 - Ejemplo de un caso armado por el agente, en base a una acción de búsqueda del usuario

3.2.3. Aprendizaje

De manera normal, al aprender una tarea *T*, utilizando alguna técnica de aprendizaje, manejamos una representación interna del dominio del problema. En nuestro caso el espacio de representación está formado por todos los perfiles posibles para un usuario dado.
Un perfil está circunscrito a su lenguaje de representación, que reúne, a su vez, los atributos del usuario y, las preferencias de este. Estos forman el espacio posible de gustos y preferencias, sobre este espacio se realiza la búsqueda del perfil del usuario. El cual debe maximizar la probabilidad de compra del cliente de productos que encajen en ese perfil.
Existen varios tipos de aprendizaje, los más importantes son, Supervisado, es decir existe un tutor que dice si lo que se hizo está bien o mal. No supervisado, sin un tutor que diga que es lo que está bien. Por refuerzo, el tutor dice si algo que se hizo estuvo bien o mal pero no explica porque y, se utiliza un cierto conocimiento a priori para luego, mediante razonamiento deductivo, inferir sus respuestas.
A su vez estos pueden ser Deductivos (mediante inferencia lógica) o Inductivos (Conocimiento parcial del dominio e inferencia)[MIT/1997] Según cuál sea la manera en que obtienen sus conclusiones.
En nuestro caso, nos encontramos ante el problema de aprendizaje por refuerzo e inductivo. Debido a que podemos saber si una sugerencia gustó o no al cliente. Pero no podemos saber siempre que atributos le gustó y cuáles no; y es inductivo, porque no estamos interesados en inferir de manera lógica que es lo que va a comprar el cliente.
Vendor aprende de la siguiente manera, observa los hábitos del usuario, analizando y, filtrando la información obtenida; y traduciéndola a casos entendibles por los algoritmos de aprendizaje, las acciones que se toman en cuenta para especificar los casos de aprendizaje y actualizar los perfiles son:

1. Cuando un producto aparece en una búsqueda
2. Cuando un producto es comprado
3. Cuando se amplía la vista de un producto (Zoom)
4. Cuando se agrega al carrito
5. Cuando una oferta no es tenida en cuenta
6. Cuando es seleccionado un producto, surgido de una búsqueda específica del usuario.
7. Se toma en cuenta el tiempo que un usuario permanece en una pagina.
8. Si la página tiene Scroll y, el usuario lo utiliza, esto también demuestra un cierto interés por lo que está viendo.
9. Cuando un usuario selecciona el color de su producto. De no haber color de su preferencia esto se utiliza también como una lección para el agente.
10. Cuando un usuario selecciona el Talle de su producto. Si abandona esta página es muy posible que no haya talles.
11. Cuando el usuario navega utilizando la barra de navegación de la aplicación. Esto le da una relevancia directa a las categorías de productos en los que el usuario está interesado.

3.2.3.1. Entorno

Para una mejor comprensión de la manera en que el agente recolecta la información, veremos con más detalle, cómo el agente confecciona los casos según las acciones del usuario. En la *Figura 3.3* se observa una de las pantallas con las que el usuario opera constantemente, cada uno de los círculos, indica una parte de la interfaz que será utilizada como fuente de información del agente. Estas se describen a continuación.

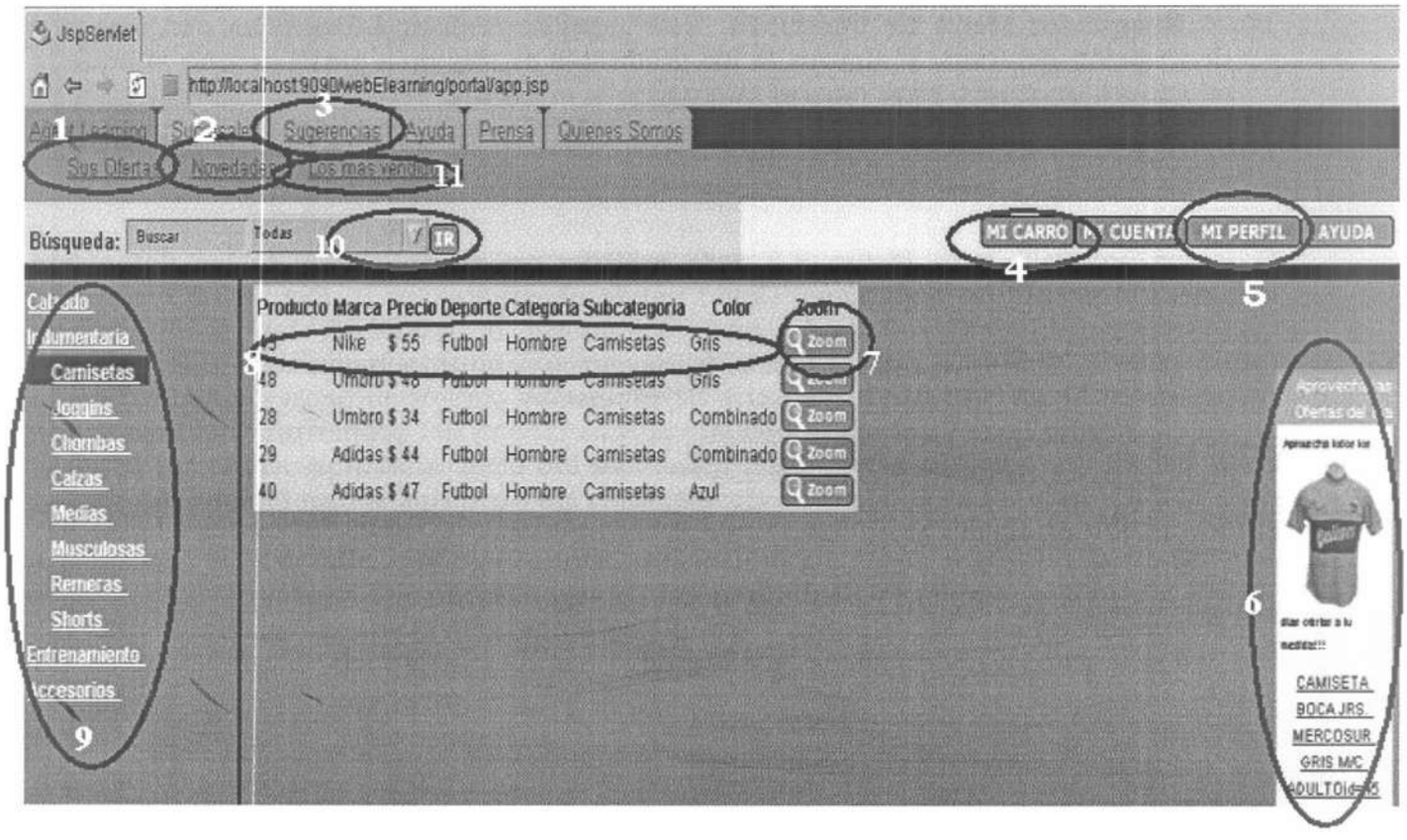

Figura 3.3 - Vista de una pantalla de la aplicación

1. **Sus Ofertas:** Por medio de esta sección se le ofrecen ofertas al usuario y, se obtiene información acerca del tipo de precio que está buscando.
2. **Novedades:** Productos más recientes y novedades. Da una idea la importancia de "Estar a la moda" del cliente
3. **Sugerencias:** Sugerencias personalizadas para el cliente. Al igual que *Sus Ofertas,* la aceptación o rechazo de estas hace que el agente aprenda nuevas lecciones.
4. **Mi Carro:** Acceso al carro de compras del cliente.
5. **Mi Perfil:** Perfil del cliente. Aquí el cliente puede personalizar el sitio. Aportando información personal, dando un poder mucho mayor al agente.
6. **Sugerencia:** En este cuadro aparecen las sugerencias hechas por el agente al usuario. Sirve para dar una idea del funcionamiento del agente, y corregir lecciones en caso de que las sugerencias no sean tenidas en cuenta.
7. **Zoom:** Vista ampliada del producto. Al hacer click aquí se muestran los detalles del producto. Que no entran en esta pantalla; y una foto del artículo, si la hay.

8. **Descripción:** Muestra algunos de los atributos más relevantes al usuario sobre los productos listados; por ejemplo, *Marca* y *Precio*.
9. **Barra de Navegación:** Barra de navegación de la aplicación, divida según las variables, Categorías y Subcategorías.
10. **Búsqueda:** Menú de búsqueda. Éste permite realizar búsquedas por marca y descripción del ítem. Si alguno de los productos es elegido a partir de una búsqueda se genera un nuevo caso para el que aprenda el agente.
11. **Los Más Vendidos:** Lista con los productos más vendidos. Éste producirá también una lección para el agente.

Podemos aprovechar la *Figura 3.3* para mostrar un ejemplo más de los datos que recolecta el agente en cada caso. Como puede observarse en la *Figura* 3.3, ya se dio una lección al agente diciendo que al usuario le gustaban los productos de Indumentaria, debido a que la barra de navegación apunta hacia esta categoría. En respuesta la aplicación trajo una serie de productos correspondientes a ese rubro. Si ahora el usuario hace Click en la opción ver más (Zoom) del primer producto mostrado en el listado, se dará un nuevo caso indicando los otros atributos *visibles al usuario;* para que aprenda a partir de estos. En este caso los atributos serían los que figuran en el caso mostrado en la *Figura 3.4*:

Usuario: PepeTrueno

Variables
Categoría: Indumentaria
Subcategoría: Camisetas
Marca: Nike
Deporte: Fútbol
Comprador : Hombre
Precio:55
Color: Gris

Fecha : 12/12/2202
Hora: 23:50

 # de Veces que realizó esta consulta: 1

Figura 3.4 - Caso de correspondiente a la pantalla de la figura 3.3

3.2.3.2. Sesgo

Con sesgo nos referimos hacia algún tipo de preferencia que pueda tener el algoritmo, para inclinarse por una solución en especial. Existen dos tipos de sesgos principales: El de búsqueda, se refiere a la elección de búsqueda en cada paso del algoritmo; y de lenguaje, se refiere al lenguaje utilizado para la representación del dominio. Vale la

pena aclarar que un algoritmo que no sesga es fútil[MIT/1997]. Debido a que el sesgo representa la inclinación hacia un tipo de soluciones. Si esta inclinación no existiera el algoritmo tendría en cuenta todo el espacio y, buscaría entre todas las soluciones posibles, lo que hace al algoritmo ineficiente, y de hecho, sin aprendizaje ya que en realidad sería un simple backtracking. Por otro lado, demasiado sesgo es malo; ya que por ejemplo podríamos caer en un agente que siempre recomendara los mismos productos. Para evitar sesgar las recomendaciones de nuestro agente **no** se volverá a mostrar un producto durante la misma sesión cuando.

1. El producto haya sido mostrado.
2. El producto haya sido comprado.
3. El producto haya sido agregado al carrito.
4. Se haya ampliado la vista del producto.

De esta manera evitamos sesgar la sugerencia de productos (sesgo de popularidad)[PAZ/2000], y hacer que un producto esté sobre expuesto. Éste criterio se utilizará en todas las implementaciones, y es ajeno a la implementación de la técnica, es decir, se aplica un nivel de abstracción más arriba que la implementación de cada técnica en particular; de esta manera simplificamos la tarea de los algoritmos de aprendizaje.

3.2.3.3. Respuesta del Usuario

No hay respuestas directas por parte del usuario, sí indirectas. Debido a que el usuario no sabe de la existencia del agente. Estas respuestas indirectas, son tomadas cuando, por ejemplo, una oferta no es tenida en cuenta, cuándo el tiempo que se queda en una página es muy largo o muy corto, si utiliza o no la barra de scroll, etcétera.
Por ejemplo, si un producto es seleccionado por el usuario, pero, al momento de elegir el talle, el usuario se arrepiente, hay que pensar que quizás los talles que hay no son de su agrado.

3.3. Asistencia al Usuario

El agente asiste al usuario mediante la sugerencia y ranking de productos, que, a criterio del agente, serán del agrado del usuario. Estas sugerencias se pueden dar a través de tres vías posibles.

1. Generación Espontánea: El agente genera ofertas automáticas para el usuario, pidiendo, por su propia cuenta, ofertas a la aplicación, o utilizando ofertas que éste posea para sugerirlas en el momento que lo crea conveniente.

2. A Pedido del Usuario: El usuario podría pedir, por propia voluntad, ver ciertas ofertas. En este caso el agente atenderá los pedidos del usuario. Utilizando ofertas que tenga de antemano, o interactuando con la aplicación para obtenerlas.

3. A pedido de la Aplicación: puede ser que la aplicación(u otro agente) decida ofertar algo al cliente, por cuenta de la aplicación misma, entonces ésta deriva las ofertas en el agente, el agente selecciona las más adecuadas, y finalmente, se las ofrece al usuario. También es posible hacer que el agente no filtre las ofertas como una manera de agregar variedad a las sugerencias.

3.4. Registro Histórico

Cada agente guardará información relativa a tiempo de ejecución, Cantidad de ofertas aceptadas y, toda aquella información que pueda ser utilizada para la comparación de los algoritmos. Esto facilitará el análisis del desempeño de cada una de las implementaciones.

3.5. Independencia de la Técnica de Aprendizaje

El agente debe ser capaz de aprender mediante distintas técnicas de aprendizaje. Si bien el objetivo de este trabajo es el de realizar una investigación sobre tres técnicas de aprendizaje específicas, la meta es poder testear muchas más, en distintos ambientes y con distintas implementaciones de agentes en varios dominios.
Por eso se desarrollo un *framework* al cual le dimos el nombre de MLF Multi Learning Framework. Éste permite parametrizar el dominio de la aplicación, y la técnica de aprendizaje; además, se encarga de manejar el registro histórico (ver la siguiente sección). A su vez cada uno de estos (técnica de aprendizaje, dominio, etcétera) puede ser un framework en sí mismo. MLF solamente se encarga de dar un marco mediante el cual poder "pegar" estas implementaciones, las cuales pueden tener interfaces altamente disímiles.
La descripción completa de MLF se detalla en al apéndice B. Damos acá un breve resumen. El framework se implementó en Java. Consta de una serie de paquetes, los cuales exportan interfaces, métodos, y clases abstractas, para los componentes que deben interactuar (agente, comercio electrónico, y técnicas de aprendizaje). Cada componente involucrado en el sistema deberá implementar o heredar, aquellas clases relacionadas con su función. De esta manera el framework brinda cohesión y comunicación entre los componentes. Luego, el framework se encarga del almacenamiento estadístico, para esto hay que implementar dónde se quiere almacenar la información, y de que manera. Con estas ideas básicas y sencillas es que se diseñó e implementó el framework.

3.6. Comentario Finales

Hemos analizado cómo funciona y cuál es la implementación de nuestro agente, explicamos cada una de las tareas que realiza, cómo aprende, cómo sugiere, y que casos tiene en cuenta mientras observa. A modo de resumen, y para lograr una idea global del funcionamiento del agente, podemos ver la en *Figura 3.5*, las tareas más importantes que realiza nuestro agente, y cómo interactúa con su ambiente.

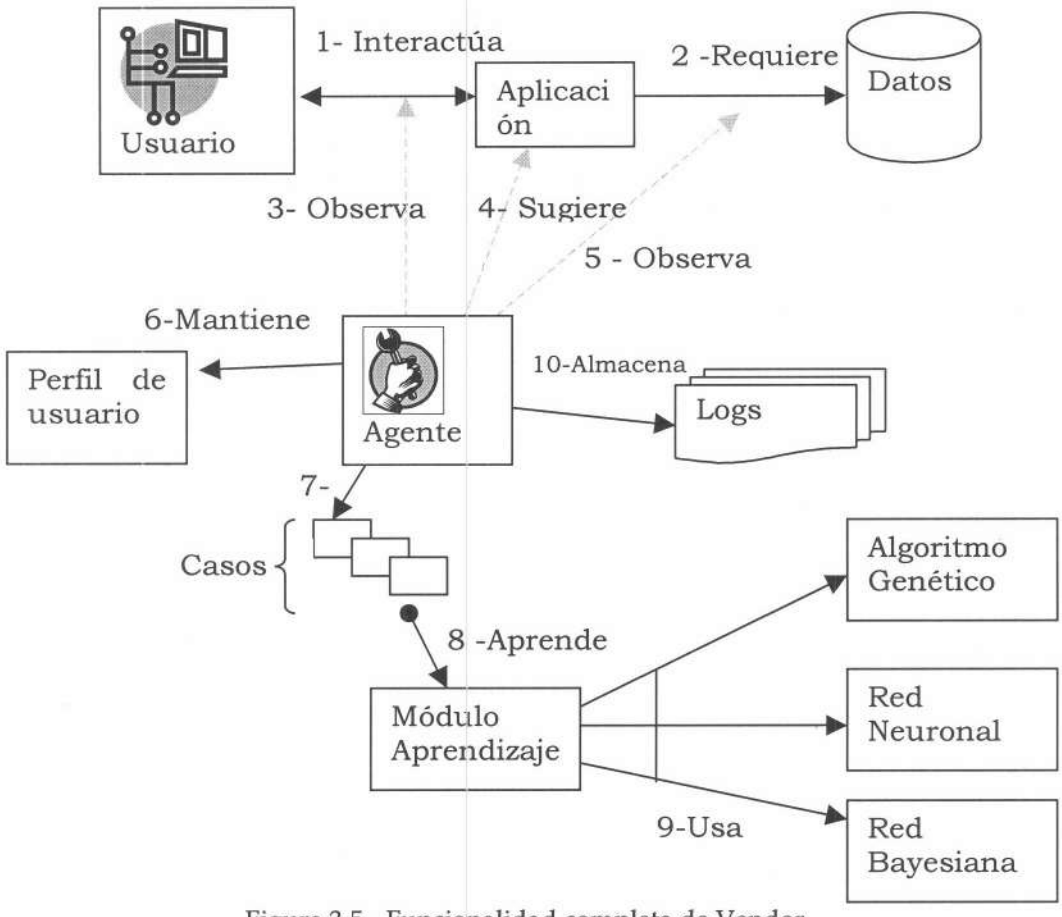

Figura 3.5 - Funcionalidad completa de Vendor

En la figura se detallan, mediante números y una leyenda descriptiva, cuáles son las acciones e interacciones, más importantes que ocurren entre, la aplicación, el agente, el usuario, y las distintas técnicas de aprendizaje. Estos se detallan a continuación.

1. **Interactúa:** El usuario interactúa con la aplicación.
2. **Requiere:** La aplicación requiere datos.
3. **Observa:** El agente Observa la interacción entre el usuario y la aplicación
4. **Sugiere:** El agente puede sugerir, por propia cuenta, productos para que la aplicación muestre al usuario.
5. **Observa**: el agente observa la interacción entre la aplicación y los datos.
6. **Mantiene:** El agente mantiene actualizado el perfil. El cual se utilizará en su momento para conocer los gustos del usuario.
7. **Genera:** El agente genera casos a partir de las observaciones realizadas
8. **Aprende:** Los casos generados son tomados por el módulo de aprendizaje, quien los utiliza para reforzar sus conocimientos.
9. **Usa:** El módulo de aprendizaje selecciona algunas de las técnicas de aprendizaje. Ya sea para aprender, o para indagar sobre algún producto.
10. **Almacena:** Mantiene un registro histórico de las consultas realizadas y los tiempos consumidos en cada una de ellas.

Una vez comprendido el gráfico, y luego de leer este capítulo, estamos finalmente en condiciones de analizar las distintas implementaciones, y los resultados obtenidos en este trabajo, temas que serán tratados en los siguientes capítulos.

Capítulo IV

4. Perfiles Utilizando Redes Bayesianas

Redes bayesianas son una poderosa arma estadística que permite representar dominios de problemas en los que intervienen muchas variables. Permitiendo representar las correlaciones (dependencias) entre las variables. Redes bayesianas han demostrado ser una importante arma estadística. Se han utilizado para modelar dominios complejos en los que existen grandes cantidades de variables y de relaciones entre ellas. Por ejemplo, se lo utilizó con éxito contundente para diagnosticar enfermedades de nodos linfáticos a partir de síntomas de los pacientes. [RUS/1995]
Vendor utilizó las redes bayesianas para modelar un perfil de usuario. Es decir las variables del perfil y su interrelación. En este capítulo explicaremos cómo se realizó esta implementación.
El capítulo se encuentra organizado de la siguiente manera. En la primera sección se presentan los conocimientos básicos sobre redes bayesianas. En la segunda parte se exponen aspectos de implementación de la red; se explica porque se eligió nuestra implementación, y la implementación en sí, y por ultimo, presentaremos algunos comentarios finales sobre este capítulo.

4.1. Redes Bayesianas

Las redes bayesianas brindan una compacta y expresiva representación de relaciones entre parámetros en un dominio [DAM/1999]. Una red bayesiana es un Grafo Dirigido

Acíclico que representa una distribución de probabilidades. Los nodos representan variables aleatorias y los arcos representan una probabilidad condicional entre las variables (Correlación). Redes bayesianas son además una muy buena herramienta para especificar dominios incompletos y manejo de incertidumbre.
En los siguientes párrafos explicaremos, mediante un ejemplo, algunos de los conceptos más importantes de redes bayesianas para poder comprender el trabajo realizado. Un análisis detallado y pormenorizado de las redes bayesianas se encuèntra fuera del alcance de este trabajo. Para esto puede referirse a [RUS/1995] y a [MIT/1997].

4.2. Representación de Conocimiento

Para representar la dependencia que existe entre determinadas variables se utiliza una estructura de datos conocida como **red bayesiana,** que sirve también, para especificar de manera concisa la distribución de probabilidad conjunta entre ellas
La red bayesiana es una gráfica que cumple las siguientes reglas:

1. Los nodos de la red están formados por un conjunto de variables aleatorias.
2. Pares de nodos pueden estar conectados entre sí mediante un conjunto de enlaces o flechas. El significado implícito de una flecha que vaya del nodo X al nodo Y es que X ejerce una influencia directa sobre Y.
3. Por cada nodo hay una tabla de probabilidad condicional que sirve para cuantificar los efectos de los padres sobre el nodo. Los padres de un nodo son aquellos nodos cuyas flechas apuntan hacia este.
4. La gráfica no tiene ciclos dirigidos (por lo tanto es un Grafo Dirigido Acíclico GDA)

De esta manera una red bayesiana representa el conocimiento que se tiene acerca de las variables así como su interrelación. Un ejemplo de estas se verá en la siguiente sección.

4.2.1. Semántica de una Red Bayesiana

Para entender la semántica de las redes bayesianas es conveniente remitirnos a algún ejemplo.[RUS/1997] Supongamos que un amigo suyo, digamos Miguel, acaba de instalar un sistema de alarmas contra robo en su casa. La detección de robos es bastante confiable, pero en algunas ocasiones, también se activa al producirse temblores. Miguel tiene dos vecinos: Juan y María, quienes prometieron llamarlo por teléfono en caso de que sonara la alarma. Juan llama siempre que suena la alarma, aunque a veces, se la confunde con el repique del teléfono, ocasiones en las que de todos modos telefonea; y María, por el contrario, disfruta de escuchar música fuerte, por lo que a veces no escucha la alarma. Con base en las probabilidades de quién llama y quién no, se quiere saber, cuál es la probabilidad qué haya habido un robo en la casa Miguel.
Una sencilla red bayesiana para este problema es representada en la *Figura 4.1*

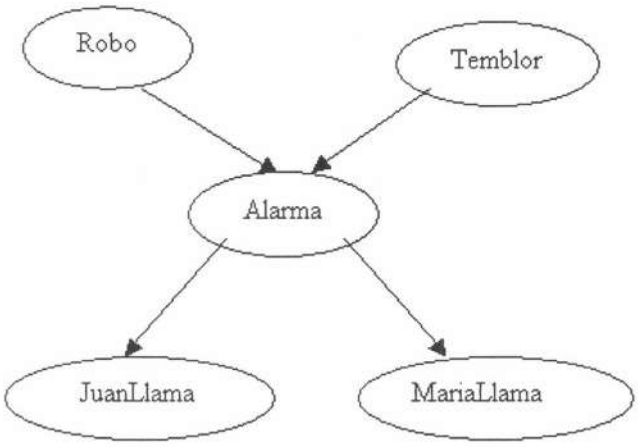

Figura 4.1 - Ejemplo de una red bayesiana sencilla

En este ejemplo la topología indica que tanto robos como temblores determinan directamente la probabilidad de que suene la alarma, pero el hecho de que Juan y María telefoneen depende, exclusivamente, de que suene la alarma. La red, por lo tanto, representa semánticamente que, ni Juan, ni María, pueden percibir directamente ningún robo, y que tampoco se dan cuenta de temblores ligeros.
Notará que en la red no hay nodos para representar a María escuchando música estridente en un momento dado, o un teléfono repiqueteando y confundiendo a Juan. Estos factores están resumidos en la incertidumbre inherente a los enlaces entre alarma y JuanLlama y MariaLlama. Lo anterior es indicio tanto de ignorancia como de pereza en la operación sobre ésta: Es mucho el trabajo necesario para saber qué haría María, para que tales factores tengan tal, o cuál posibilidad, de incurrir en un caso particular; y por otra parte, no existe una manera razonable que nos permita obtener la información correspondiente. De hecho, las probabilidades resumen un conjunto de posibles circunstancias, en principio infinito, en las que la alarma pudiera no sonar (Exceso de humedad, falta de energía eléctrica, etcétera), o bien, que Juan o María, no telefoneen, ni notifiquen (salieron a comer, están de vacaciones, etcétera). Es decir una pequeña red tiene que vérselas con un mundo inmenso, y debe hacerlo, al menos, de manera aproximada.
Una vez especificada la topología, hay que hacer lo mismo con la tabla de probabilidad condicional, de cada uno de los nodos. Cada hilera de la tabla indica la probabilidad condicional correspondiente al valor de un nodo de un caso condicionante. Un caso condicionante es una posible combinación de los valores de los nodos padres (un evento atómico en miniatura, si así desea verlo). Por ejemplo, la tabla de probabilidad condicional de la variable aleatoria alarma sería la siguiente.

B	E	P(ALARMA | ROBO, TEMBLOR) VERDADERO	FALSO
Verdadero	Verdadero	0.95	0.050
Verdadero	Falso	0.95	0.050
Falso	Verdadero	0.29	0.710
Falso	Falso	0.001	0.999

Tabla 4.1 - Tabla de probabilidad condicional para la variable alarma.

La suma total de las hileras debe dar 1, puesto que las entradas representan un conjunto exhaustivo de los diversos casos de una variable. Por ello, sólo uno de los números que aparecen en las hileras, se puede especificar de manera independiente. En general, en una tabla para variables booleanas que contengan padres booleanos habrá 2^N probabilidades que se pueden especificar de manera independiente. Un nodo sin padres tendrá sólo una hilera, la cual representa las probabilidades anteriores de cada uno de los posibles valores de la variable.

En la *Figura 4.2* se muestran las tablas de probabilidades correspondiente al ejemplo de la alarma. En esta figura se muestra solamente la probabilidad condicional correspondiente al caso *Verdadero* de cada uno de las variables (el valor de Falso es igual a 1-el valor de *Verdadero*) Las distintas probabilidades de la *Figura 4.3* son:

P(B) = P(robo) Probabilidad de que ocurra un robo.
P(E) = P(Temblor) Probabilidad de que ocurra un temblor.
P(A) = P(alarma/robo, Temblor) Probabilidad de Alarma dado Robo y Temblor.
P(J) = P(JuanLlama) Probabilidad de que Juan Llame con Alarma como evidencia.
P(M)= P(MariaLlama) Probabilidad de que María Llame con Alarma como evidencia.

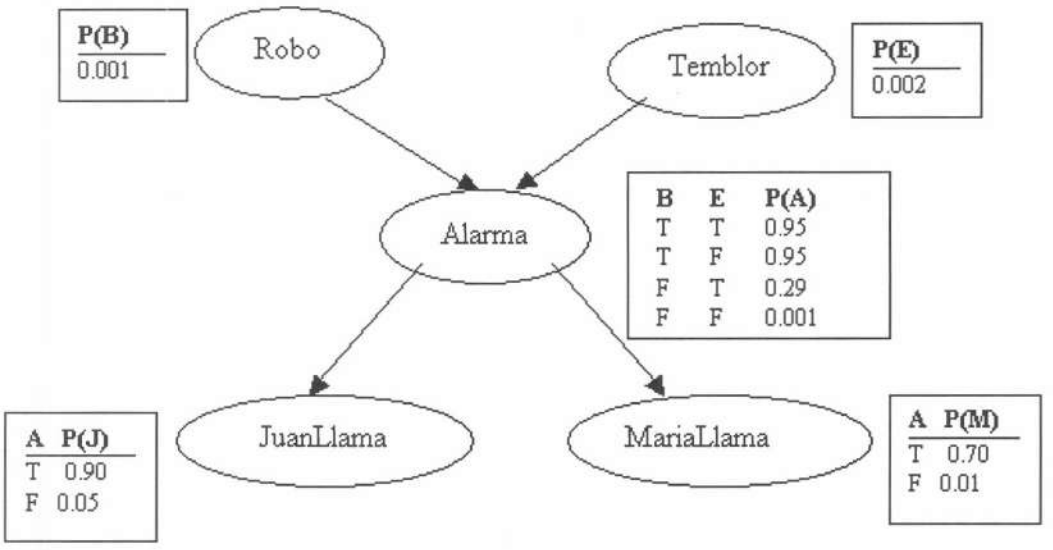

Figura 4.2 - Probabilidad para el ejemplo de la Alarma

Debido a que podemos especificar todas las relaciones entre las variables, es que(como dijimos anteriormente) la red de creencia permite dar una descripción completa del dominio. Es posible calcular cada una de las entradas de distribución de probabilidad conjunta desde la información de la red. Una entrada genérica en la probabilidad conjunta es la probabilidad de que se dé la conjunción de determinadas asignaciones para cada una de las variables, como sería P(X1=x1 ∧ X2=x2 ∧ .... ∧ Xn=xn) Usualmente se utiliza la notación P(x1,....,xn) como abreviación de la anterior. El valor de esta entrada viene dado por la siguiente formula:

$$P(x_1,..x_n) = \prod_{i=1}^{n} P(x_i \mid Padres(x_i))$$

Figura 4.3 - Cálculo de una entrada de probabilidad

Por lo tanto, las entradas de la probabilidad conjunta se representan mediante el producto de los respectivos elementos de las tablas de probabilidad condicional (TPC) de la red bayesiana. Es decir, las TPC constituyen una representación analítica de la probabilidad conjunta.

4.2.2. Como Inferir un Determinado Valor

Mostraremos con un pequeño ejemplo cómo funciona la ecuación dada en la *Figura 4.3.*En el ejemplo queremos calcular la probabilidad de que suene la alarma sin que se haya producido robo, ni temblor, habiendo llamado tanto Juan como María. Las variables se presentarán mediante una sola literal. El resultado de este cálculo se muestra a continuación.

$$P(J \wedge M \wedge A \wedge \neg B \wedge \neg E) = P(J \mid A)P(M \mid A)P(A \mid \neg B \wedge \neg E)P(\neg B)P(\neg E) =$$
$$0.90 * 0.70 * 0.001 * 0.999 * 998 = 0.00062$$

4.2.3. Tipos de Problemas

Según la topología de la red utilizada, y según cuál sea el conocimiento de las variables que intervienen en el dominio podemos clasificar los distintos tipos de problemas conocidos en redes bayesianas.
El problema del aprendizaje en las redes bayesianas se manifiesta de diversas maneras. Es posible que sea conocida o desconocida la estructura de la red y las variables de dicha red pueden ser observables u ocultas. Por lo tanto podemos derivar fácilmente la siguiente taxonomía.

4.2.3.1. Estructura Conocida Completamente Observable

En este caso, la única parte de que se puede aprender es el conjunto de las tablas de probabilidad condicional. Esto puede ser calculado directamente a partir de las estadísticas del conjunto de ejemplos. En alguno sistemas de redes bayesianas se cuenta con la actualización automática de las entradas de la tabla de probabilidad condicional, para reflejar los casos que van conociéndose.

4.2.3.2. Estructura Desconocida Completamente Observable

En este caso el problema reside en reconstruir la topología de la red. Este problema puede ser considerado como una búsqueda a través del espacio de estructuras, guiado por la capacidad de cada una de las estructuras para modelar adecuadamente los datos.

4.2.3.3. Estructura Conocida Parcialmente Observable

Este caso es similar al aprendizaje en redes neuronales. (Más información sobre el tema puede encontrarse en el capítulo 6) en que, conocemos la estructura pero no sabemos que pasa con los nodos intermedios.

4.2.3.4. Estructura Desconocida Parcialmente Observable

Cuando algunas de las variables resultan inobservables, siempre o en ciertas ocasiones, se dificulta la aplicación de las técnicas de probabilidad anterior para recuperar la estructura, ya que básicamente necesitan promediar todas las posibles combinaciones de valores de las variables desconocidas. Hasta el momento no se dispone de buenos algoritmos generales para resolver este problema.

4.3. Redes Bayesianas Adaptativas

Se conoce como redes adaptativas aquéllas que cambian las probabilidades de sus nodos de manera paulatina, con cada ejemplo, para adaptarse al ambiente. Un ejemplo

de estas redes, puede ser, las redes bayesianas que utilizan algoritmos tipos backpropagation[RUS/1995] para establecer el peso de sus nodos.
Una red adaptativa es lo que necesitamos para aprender un perfil de usuario. Debido a que puede adaptarse en el tiempo a los gustos del usuario. Esto se explicará más adelante en la siguiente sección.

4.4. Propiedades de las Redes Bayesianas

Caracterizaremos a las redes bayesianas desde el punto de vista del aprendizaje de máquina, identificando sus puntos fuertes, tipos de problemas para los que son apropiados, y configuraciones típicas para distintos tipos de problemas. De esta manera podemos estimar que tan buenas pueden resultar en la tarea de mantener y adaptar un perfil de usuario.[SCH/2001]

4.4.1. Sesgo Inductivo

Sesgo de lenguaje de representación:
El sesgo viene dado por la representación de atributos discretos, para los cuales deberá asociarse la probabilidad de su ocurrencia, y principalmente por la asignación de los valores de probabilidad con los cuales se cree la red.[MIT/1997]

Sesgo de la Búsqueda
El espacio de hipótesis del algoritmo se encuentra formado por todos las posibles valores de probabilidad y combinaciones de éstos, que pueda tomar la probabilidad de *P.* Donde cada una de la hipótesis se corresponde con una asignación de valores de probabilidades para cada *P.* [MIT/1997]

4.4.2. Problemas Apropiados

- En donde existe conocimiento inicial sobre algunas probabilidades.
- Cuando el problema tiene información sobre probabilidades (en general predicciones sobre la instancia).
- Cuando la función a representar es probabilística

4.4.3. Ruido

Robusto al ruido por estar formado por estimaciones probabilísticas. Esto hace que en vez de oscilar con el ruido, lo tenga en cuenta de manera probabilista. [RUS/1995]

4.4.4. Generalización de Nuevos Casos

Es bueno ya que al no eliminar hipótesis inconsistentes utiliza estas para corregir las suposiciones estadísticas. [MIT/1997]

4.4.5. Sobreaprendizaje

Por ser una técnica probabilista es muy robusta al sobre aprendizaje, siempre y cuando, se lo tenga en cuenta en las probabilidades. Es decir las redes bayesianas son

robustas al sobreaprendisaje debido al valor probabilista de los hechos, estos valores deberán variar lentamente para no caer en sobreaprendisaje. Por ejemplo: si asignáramos a un hecho que muy rara vez ocurre, y que en nuestros ejemplos nunca ocurrió, la probabilidad de que no ocurra el valor 1, estaríamos haciendo que la red sobre aprendiera, porqué el hecho de que nunca nos hallamos topado con un caso así, no quiere decir que no exista, por lo tanto una probabilidad del tipo 0.999998, o cualquier otro menor a 1, evitará que la red sobre aprenda un concepto erróneo.

4.5. Redes Bayesianas Para Generar Perfiles Dinámicos de Usuarios

Las redes bayesianas brindan una compacta y expresiva representación de relaciones entre variables en un dominio [DAM/1999]. Una red bayesiana es un Grafo Dirigido Acíclico (GDA) que representa una distribución de probabilidades. Los nodos representan variables aleatorias y los arcos representan una probabilidad condicional entre las variables que une (Correlación). Tablas de probabilidad condicional especifican de manera cuantitativa la probabilidad. Es decir, para cada nodo, existe una tabla de probabilidad con la probabilidad de cada posible estado del nodo dada cada posible probabilidad de sus padres. La tabla para los nodos raíces (sin padres) simplemente contienen la probabilidad incondicional de las variables[HAD/1999].
Si alguno de estos conceptos no le es familiar es conveniente leer la sección 1 de este mismo capítulo. En el cual se describe las redes bayesianas.
Las redes bayesianas permiten modelar las relaciones que ocurren entre los atributos de un producto y un perfil de usuario. Son una poderosa herramienta de predicción debido a varias razones, entre ellas, permite representar un dominio completo, es decir, todas las relaciones entre los atributos. Permite tener una representación del dominio que, no solamente infiere un determinado resultado con una probabilidad dada, sino también, permite dar una explicación (o varias) del por qué de esta inferencia. Para implementar el aprendizaje bayesiano se utilizó JavaBayes[JAV/01]. JavaBayes es un framework para el desarrollo de redes bayesianas.

4.5.1. Representación de Perfiles por Medio de una Red Bayesiana

Sabemos que el dominio del comercio electrónico está compuesto mayoritariamente por atributos discretos; y que muchos, guardan cierta correlación entre ellos, correlación que es, en muchos casos, intuitiva y fácil de expresar por expertos de dominio. Ejemplo de este tipo de correlación podría ser la dependencia entre las variables *sexo* y *deporte*, donde posibles elementos serían (Hombre, Fútbol), (Mujer, Hockey), (Hombre, Golf). De esta manera, podemos pensar un perfil en sí mismo como la distribución de probabilidades de una red bayesiana que representa las probabilidades asociadas a cada categoría y la relación entre ellas.
Utilizando estas ideas cualquier arquitectura de red que elijamos respeta los siguientes conceptos.

1. Cada nodo representa una variable del dominio del calzado, por ejemplo Precio.

2. Los valores posibles del nodo representan los valores posibles de la variable. Para el caso del nodo precio, estos valores pueden ser: Oferta, Normal, SobrePrecio.
3. Los nodos raíces (Periféricos) sólo contienen la tabla de probabilidades de los posibles valores que puede tomar el nodo.
4. Cada Arco entre dos nodos representa la correlación entre esas dos variables.
5. Para cada Nodo interior (que tiene un padre, al menos) se especifica la probabilidad de cada posible valor dada la probabilidad de cada combinación de valores de sus padres.

La cantidad de nodos mínima está clara, habrá, al menos, uno por cada variable. Ahora bien nos falta saber cuál será la estructura de la red, esto es, qué nodos se relacionan mediante arcos. Cuestiones de diseño y actualización de la red serán explicadas más claramente en la siguiente sección.

4.6. Arquitectura de la red

La estructura de la red se armó sobre la base del análisis de los datos (Ver apéndice B para más detalle). Además, tuvimos en cuenta la opinión de un experto en el dominio.
La red se armó de la siguiente manera: Se agruparon los nodos según los atributos más importantes de éste. Para esto se agrupan los nodos que guardan una cierta relación entre sí, formando los siguientes nodos: *Moda, CaracteristicasPersonales* y *Fecha.* Se tomo en cuenta que tanto contribuía cada atributo del dominio (*Color, Talle, Precio,* etcétera) con cada uno de los nuevos nodos. Con esta arquitectura buscamos desacoplar la red bajando la cantidad de relaciones entre atributos. Ahora las relaciones se darán a través de los nodos agregados.
Se relacionó el atributo *marca* y el atributo *Deporte* debido a que estos tienen mucha importancia, y tienen bastante que ver entre sí. (hay una fuerte correlación)
En la *Figura 4.4*, se muestra el gráfico de la red con los nodos intermedios y los atributos que agrupa cada uno. Los nodos intermedios aparecen observados en azul.

Detallaremos a continuación cómo se agrupan los nodos

Fecha Importante capta la importancia de la fecha, consta de los días {lunes a vienes}, semana {primera a cuarta} y mes {enero a diciembre}

ModaPrecio: Agrupa la *Calidad*, el *Tiempo* de Exposición y el *Ranking* de venta interno de los productos.

CaracterisiticasPersonales: Agrupa *Talle, Color,* y *Target*

MarcaDeporte: Agrupa la *Marca* el *Deporte* y la *Categoría,* debido a que en Gral. determinadas categorías están más relacionas con determinadas marcas. (ejemplo marca Flash y categoría Accesorios, o Marca Nike y Categoría Indumentaria)

ImportanciaGral: que agrupa los nodos *CaracterisiticasPersonles* y *Marca_Deporte.*

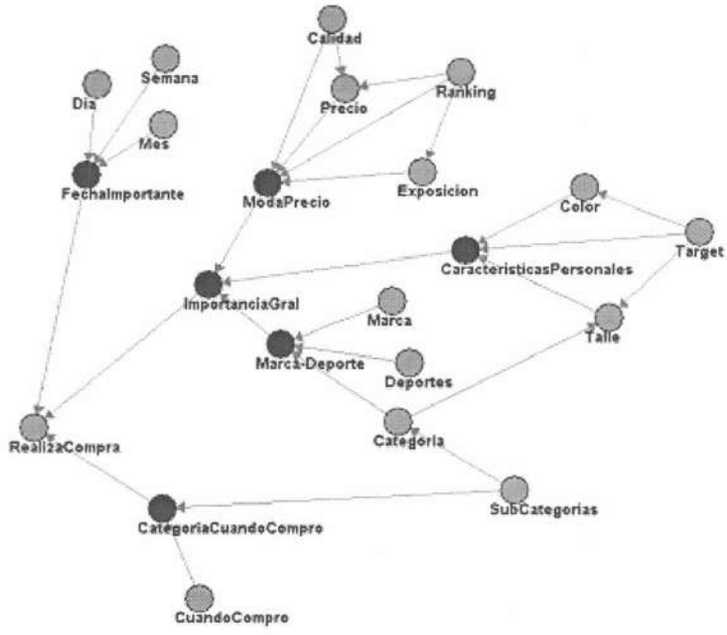

Figura 4.4 - Diseño de la red bayesiana utilizada por Vendor

CategoriaCuandoCompro: que indica la probabilidad de que un usuario compre un articulo de una determinada categoría dado que hace X tiempo compró un artículo de la misma categoría, es decir es difícil que un usuario vuelva a comprar zapatillas si hace apenas dos semanas compró zapatillas. Este nodo trata de captar esa probabilidad.
Por ultimo, *RealizaCompra*: que agrupa todos los atributos intermedios, y tiene el fin de captar el interés del usuario. Es decir, si los atributos del producto y la fecha son buenos la probabilidad de *RealizaCompra* será alta.

Las demás relaciones son relaciones existentes entre los atributos, que se implementaron para poder realizar otras consultas interesantes. (inferencias causales)

4.7. Modo de Actualización

Existen varias formas de actualizar una red bayesiana cuya estructura es conocida, y sus variables completamente observables, como es nuestro caso. Una de ellas es calcular las probabilidades condicionales cada vez. Otra forma es realizar algún algoritmo del tipo descenso de gradiente o backpropagation (Redes conocidas como RPA – Redes Probabilistas Adaptativas).[RUS/1995]
Optamos por la segunda, debido a que es la más rápida; ya que, para calcular las probabilidades cada vez, habría que realizar varias consultas a una base de datos, situación que llevaría más tiempo que la consulta misma que está ejecutando para atender el pedido del usuario, y esto no es deseable para la performance del agente.

Por esto se implementó un algoritmo para realizar la adaptación. Este algoritmo funciona de la siguiente manera.
Cada nueva lección produce la adaptación de la red. Cada caso viene dado por, al menos, el valor de un atributo, al que se debe modificar su probabilidad, en detrimento de los otros.
Por ejemplo, teniendo en cuenta la estructura de red que figura más arriba (*Figura 4.4*) y, los dominios de los atributos que definimos en el capítulo II. Supongamos que deseamos modificar el valor del elemento *Hombre* de la variable *Target.* Esto quiere decir que hay que darle una mayor probabilidad al elemento *Hombre,* respecto de los demás valores dentro de la categoría. En las *Tablas 4.2 y 4.3* se ven las tablas de probabilidad condicional de la variable, antes y después de modificar su valor, respectivamente. Los parámetros que se muestran en las figuras son:

- *Ni* Es algún valor posible de la variable involucrada.
- δ Es el orden de crecimiento para un elemento.
- $\delta/(\#Variable - 1)$ Es el orden de crecimiento, divido, la cantidad de Elementos de la Variable menos uno. En este caso es 4 (*Niño, Mujer, Adolescente, Unisex).* La idea que el crecimiento de un elemento influya proporcionalmente en los elementos.
- En cualquier caso debe pasar que:
  - $(Ni + \delta) <= 1$
  - $(Ni - (\delta/(\#Variable - 1))) >= 0$

Valores	Probabilidad
hombre	N1
Niño	N2
mujer	N3
adolescente	N4
unisex	N5
Total	**1=(N1+N2+N3+N4+N5)**

Tabla 4.2 - Probabilidad inicial de los elementos de la variable *target*

Valores	Probabilidad
Hombre	N1 + δ
niño	N2 - δ/4
mujer	N3 - δ/4
adolescente	N4 - δ/4
unisex	N5 - δ/4
Total	**1=(N1+N2+N3+N4+N5)**

Tabla 4.3 - Probabilidad final de los elementos de la variable *target*

4.7.1. Ejemplos

Daremos ahora dos ejemplos que muestran cómo funciona el algoritmo en casos con más de una variable. Luego mostramos el pseudocódigo de la implementación de este.

4.7.1.1. Primer Ejemplo

Supongamos que hacemos un click en la barra de navegación en el *Categoría calzado* y en la *subcategoría tenis*.
Esto generará que la red bayesiana aprenda una lección y es qué la categoría *Calzado* y la subcategoría *Tenis* tienen más importancia que antes. El caso correspondiente a esta lección es el de la *Figura 4.5*.
Las Variables involucradas en el caso son: *Categoría, Subcategoría, CuandoCompro y Fecha* (la cual se traduce a las variables *Dia, Mes, Semana*).Viendo la estructura de la red, tenemos que, en la primer iteración los únicos atributos que no tienen dependencias (esto quiere decir que no tienen flechas entrantes, por ende, el valor de sus Tablas de probabilidad no dependen del valor de otra variable) y que pueden observarse son: *Categoría, subcategoría, CuandoCompro, día, mes, y semana.* De esta manera, con los datos que tenemos en el caso, deberíamos observar las variables que figuran en éste; e incrementar la importancia de cada una de ellas (aumentar la probabilidad de que ocurra). la red quedaría como se muestra en la *Figura 4.6*

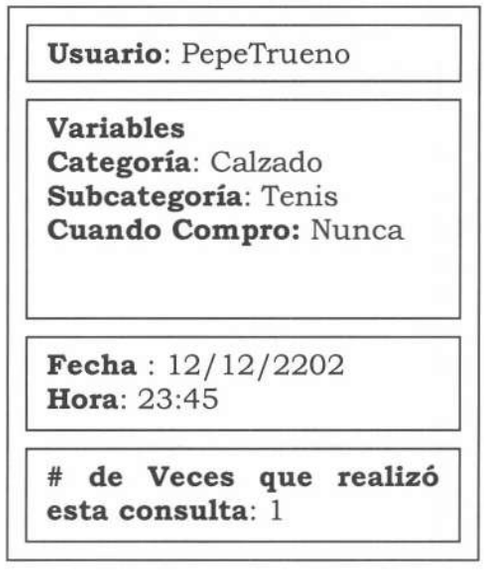

Figura 4.5 - Caso a aprender por la red bayesiana

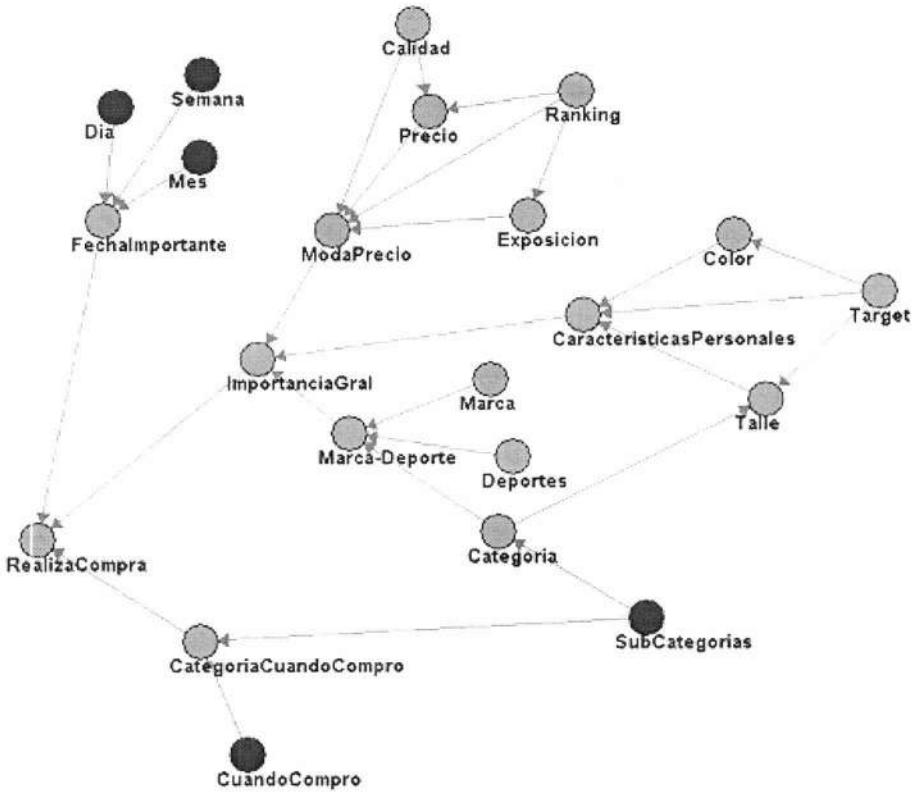

Figura 4.6 - Red observada con los valores de los atributos del caso

Ahora, si hacemos una segunda iteración, el nodo *CategoriaCuandoCompro* no tendrá dependencias, esto quiere decir que sus padres están observados, y como hay un valor para esta variable en el caso, se la puede observar; y aumentar la probabilidad de que sea probable la compra. *Figura 4.7*
Es decir:

P(CategoriaCuandoCompro=probable/ subcategoria=Ind_CASUAL∧ CuandoCompro=nunca)

Lo mismo pasa para el atributo *FechaImporante.*

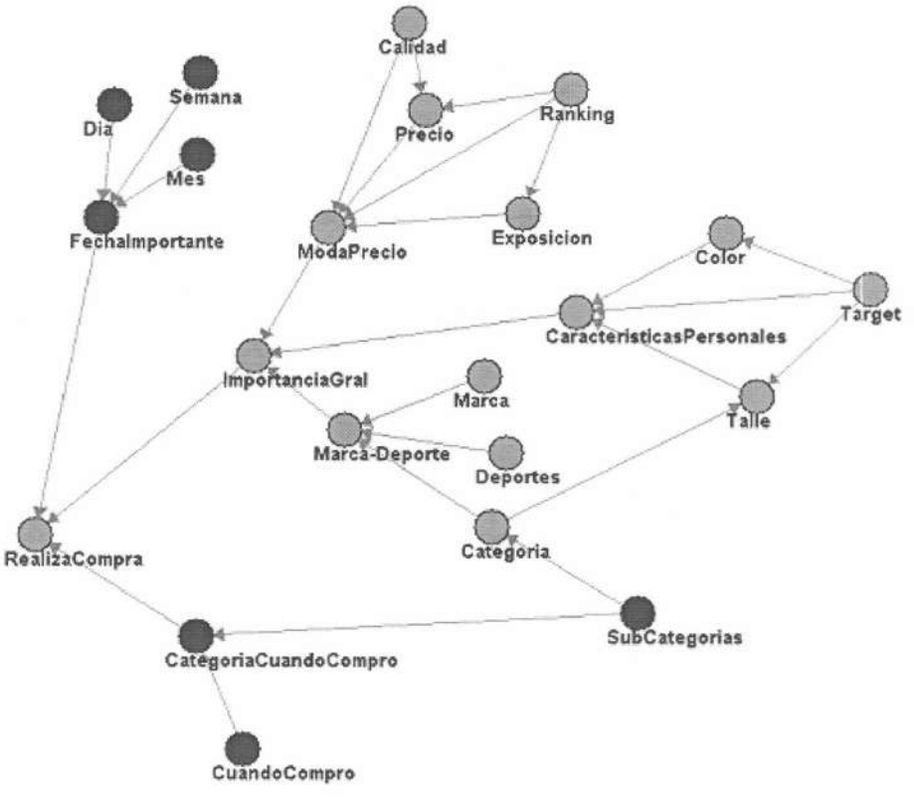

Figura 4.7 - Caso a aprender por la red bayesiana

En la próxima iteración vemos que no hay ninguna otra variable que no tenga dependencias, debido a que falta conocer atributos de ellas, o de sus padres, para observar otros nodos; por ejemplo, falta el *Target* para poder observar el *Talle.*

4.7.1.2. Segundo Ejemplo

Supongamos que realizamos la compra de un producto. Ahora el caso dispone de todas las variables raíces para su utilización.
En ese caso la red en la primera iteración quedara como se muestra en la *Figura 4.8*, Con todos los atributos periféricos observados.

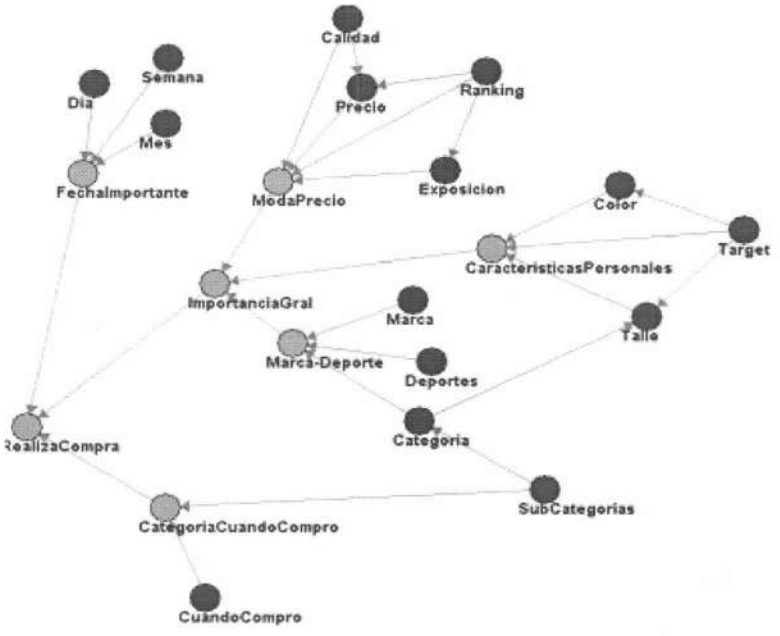

Figura 4.8 - Algoritmo de actualización -Primera iteración

Debido a que ahora el usuario conoce el producto y por ende podemos observar todos los atributos.

Luego en la segunda iteración tendremos que se asignan los siguiente atributos.

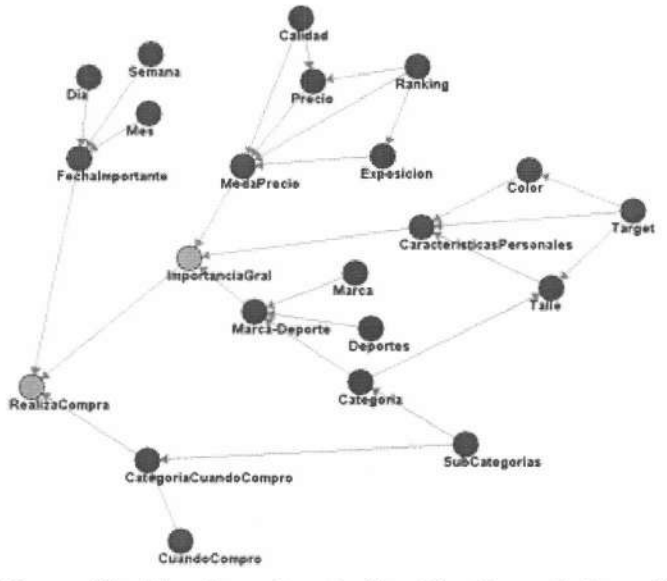

Figura 4.9 - Algoritmo de actualización -Segunda iteración

En la tercera.

Se observará el atributo ImportanciaGral

Por ultimo.

Se observará el atributo realiza compra.

Cabe destacar que el orden de crecimiento(δ) varia según cuál sea la transacción que realiza el usuario en la red, es decir si amplía el producto el ratio será uno, pero si compra el producto el ratio será mayor.
Así mismo en cada momento se utilizan como atributos conocidos, propios del producto, aquellos que son visibles por el usuario al momento de la transacción, es decir, si el usuario no sabe en que color está disponible el producto, entonces el atributo color, no es tenido en cuenta para la actualización de la red.
Los resultados de esta implementación se verán en el capítulo VII, en el cual se realiza la comparación de las distintas implementaciones de nuestro agente.

4.7.2. Algoritmo

El algoritmo pertenece a la clase de condicionamiento, debido a que se observan los valores de los distintos nodos de manera iterativa, hasta formar un poliarbol que será evaluado y modificado, cambiando así los valores de las variables. [RUS/1995].
El algoritmo comienza con una red la cual no tiene ninguna variable observada. Luego recorre cada una de las variables del caso, viendo si estas tienen dependencias, si no las tienen se observa la variable y se incrementa la importancia del elemento. El algoritmo itera hasta encontrar un punto fijo, es decir, una iteración en la que la red sigue igual a la iteración anterior.
El pseudocódigo del algoritmo se describe a continuación.

```
Funcion aprender ( RedBayesiana red, double OrdenCrecimiento, Caso caso))
{
/**
caso: Es un conjunto de atributos
OrdenCrecimiento: Es el orden de crecimiento de los valores a observar.
Red: Es la red Bayesiana
*/
//no HayPuntoFijo si realicé al menos una modificación al caso en el ciclo interior del mientras//

Mientras (Caso.noVacio() && noHayPuntoFijo()){
        Para Cada Variable v Ɛ caso )
        {
                si (noTieneDependencias(variable))
                {
                        setearProbabilidad(ordenCrecimiento, variable, elemento);
                }
        }
}
}
Funcion booleano noTieneDependencias(variable, valor)
{
        si (todos los nodos predecesores de variable están observados, o no hay nodos predecesores)
                retornar Verdadero;
        sino
                retornar Falso;
}
Funcion setearProbabilidad(double ordenCrecimiento, Variable v, String elemento)
{
/**
Básicamente aumenta la posibilidad de una probabilidad y disminuye lo aumentado de manera proporcional a
las otras probabilidades
**/
        doble probabilidadActual = v.getProbabilidad(elemento);
        doble probabilidadFutura = probabilidadActual + ordenCrecimiento;
        si (probabilidadFutura <=1 and probabilidadFutura >=0)
        {
                v.setProbabilidad(elemento, probabilidadFutura)]
        }
        doble ordenOtros = ordenCrecimiento /c.cantidadElementos()
        para cada Elemento e Ɛ v and e !=elemento
        {
                doble actual = v.getProbabilidad(e);
        doble futura =actual + ordenOtros;
        v.setProbabilidad(e, futura)
}
}
```

Figura 4.10 – Pseudocódigo del algoritmo de actualización

4.7.2.1. Parámetros Utilizados en Cada Lección

Como puede verse, lo que indica la velocidad de adaptación de la red es la variable *OrdenCrecimiento* marcando que tan bruscamente y cómo se modifican los atributos, en este momento algunos, son valores fijado por un experto de dominio, y otros fueron modificándose para obtener resultados coherentes. Por ejemplo la importancia de algo

que busca el usuario debe ser mayor que la importancia de algo que aparece en la barra de navegación. Una posible mejora del algoritmo sería establecer algún orden sobre la base de la cardinalidad del dominio del atributo, o al de la función. Así como también podría tener en cuenta la longitud desde el atributo hacia el nodo destino (RealizaCompra) en el grafo formado por la red.

4.8. Comentarios Finales

En este capítulo vimos una pequeña reseña sobre redes bayesianas y su utilidad para generar perfiles dinámicos de usuarios. Mostramos cómo podíamos representar un perfil utilizando redes bayesianas. Se muestra también la elección de la arquitectura de la red. La representación de las variables y las dependencias entre ellas. Se explica el modo en que actualizamos la red y el algoritmo desarrollado para esto; y se dan varios ejemplos sobre la actualización.
Para ver cuál fue el resultado obtenido, y los casos de test de esta implementación, es conveniente pasar al capítulo VII en el cual se realiza un análisis del desempeño de esta implementación del agente.
Queda como un trabajo futuro analizar cuál es la mejor forma de actualizar la red; ya sea, probar utilizando algún otro tipo de algoritmo backropagation; quizas uno más puro, en el sentido teórico de este tipo de algoritmos; o la modificación del orden crecimiento actual, el cual es una formula sencilla pero eficiente, por alguna otra un poco más compleja, pero que deberá ser también más eficiente.

Capítulo V

5. Perfiles Utilizando Algoritmos Genéticos

Los algoritmos genéticos son una técnica que busca resolver problemas de búsqueda tomando como idea la evolución de las especies. Algunas de las características conocidas de los algoritmos genéticos son, su robustez y eficiencia, para resolver problemas en los que interaccionan gran cantidad de atributos. Otro atributo inherente de la técnica, es la capacidad de adaptación en el tiempo[MIT/1997]. Debido a estas características, es importante realizar un análisis de la eficiencia de éstos en la generación y aprendizaje de perfiles dinámicos de usuario.

Si bien existen trabajos anteriores que utilizan algoritmos genéticos para la generación de perfiles de usuarios [SOR/1995] y [BEE/1994], pocos utilizan esta técnica de manera pura. En general, sólo se utiliza la técnica para agregar diversidad a la población. En este trabajo, no sólo analizaremos la técnica pura, sino también, analizaremos varias posibles configuraciones del algoritmo.

El capítulo se encuentra organizado de la siguiente manera. En la primera sección se presenta una breve reseña de algoritmos genéticos, y los conceptos fundamentales para entender el trabajo realizado. En la segunda parte se presenta la implementación de algoritmos genéticos para *Vendor,* y se explica en detalle, las distintas configuraciones e implementaciones que se tuvieron en cuenta. Por último realizamos algunos comentarios finales sobre el trabajo realizado.

5.1. Algoritmos Genéticos

Los algoritmos genéticos son una técnica metaheurística para la solución de problemas de optimización. Una metaheurística es un proceso iterativo que guía a una heurística subordinada, combinando, inteligentemente, diferentes conceptos, para explorar y explotar el espacio de búsqueda, usando estrategias de aprendizaje para estructurar la información, con el objetivo de encontrar, eficientemente, soluciones cercanas al valor óptimo[OSM/1995].

Los algoritmos genéticos se basan en la analogía de la teoría biológica de la evolución de las especies, y toman esta idea, para buscar una o más soluciones optimas entre un conjunto de posibles soluciones.

La teoría de la evolución explica el origen y transformación de las especies de los seres vivos, como el producto de la acción, de dos principios fundamentales: La selección natural, y el azar. La Selección natural, regula la variabilidad de la recombinación y mutaciones aleatorias de los genes: Toda la variedad que observamos en la naturaleza viviente, se basa en la capacidad de los seres vivos de producir copias de sí mismos. En el proceso de reproducción actualiza muchas variantes, y en la interacción con el ambiente, algunas de ellas son seleccionadas para sobrevivir, y producir las copias subsiguientes[GAR/2000]

Algoritmos genéticos generan sucesores de los actuales individuos, mediante la recombinación (cruza) y mutación de partes, de las mejores hipótesis actuales (población). De esta manera, la población evoluciona de manera iterativa. Quedando y obteniendo los individuos que mejor se adaptan al mundo; es decir, los más cercanos al óptimo

5.2. Anatomía de un Algoritmo Genético Simple

Los algoritmos genéticos exploran un espacio de soluciones candidatas, en busca de la mejor solución. Cuando decimos la mejor solución, nos referimos a aquélla que optimice cierta función relevante para el problema tratado, a esta función se la conoce con el nombre de fitness o función de aptitud.

A pesar que existen clases muy diversas de algoritmos genéticos, todas mantienen una estructura común. En cada iteración, todos lo miembros de la población son evaluados de acuerdo a la función de fitness. Una nueva población es generada, seleccionando en forma probabilista los mejores individuos de la población actual. Algunos de estos individuos son pasados a la próxima generación automáticamente, mientras que otros, son utilizados para procrear soluciones por medio de cruzamientos de dos individuos, o bien son mutados, antes de pasar a la próxima generación.

El algoritmo itera hasta cumplir con un criterio de parada. Éste puede ser la cantidad de iteraciones realizadas, o la obtención de una solución que esté dentro de un cierto umbral de aceptación.

El pseudocódigo del algoritmo básico para un algoritmo genético simple se muestra en la *Figura 5.1* [MIT/1997]

Algoritmo *Genético*
input: f función de fitness,
f_umbral criterio de terminación,
p cantidad de individuos de una población,
r probabilidad de cruce,
m probabilidad de mutación
output: individuo con mayor f

Inicialización de la población: $P \leftarrow$ Generar p individuos al azar
Evaluación: **for each** h **in** P **do**
Calcular $f(h)$
Ciclo principal: **while** $([\max_h f(h)] < f_umbral)$ **do**
Crear una nueva generación, P_S*:*

Selección: Seleccionar probabilísticamente $(1 - r)p$ miembros de P para agregar a P_S

Cruzamiento: Seleccionar con probabilidad r pares de individuos y producir descendencia aplicando el operador de cruce y agregarlos a P_S

Mutar: Seleccionar con probabilidad m individuos de P_S. Para cada uno utilizar el operador de mutación y agregar el individuo modificado a P_S

Actualizar: $P \leftarrow P_S$

Evaluación: **for each** h **in** P **do**
Calcular $f(h)$

return individuo con mayor f

Figura 5.1 - Esqueleto de un algoritmo genético

5.2.1. Codificación de los Cromosomas

Manteniendo una cierta analogía con la naturaleza, podemos decir que, los individuos están formados por Cromosomas. Los cromosomas determinan cómo será el individuo, el cual depende del problema a resolver. Por eso mismo, la representación de los cromosomas depende también del problema a tratar, en nuestro caso veremos varias de ellas, esta elección influye directamente sobre los resultados del algoritmo genético. Los algoritmos genéticos requieren que el conjunto se codifique en un cromosoma. Cada cromosoma tiene varios genes, que corresponden a parámetros del problema. Para poder trabajar con estos genes en una computadora, es necesario codificarlos en una cadena, es decir, una tira de símbolos (números o letras).

Existen distintos esquemas generales de codificación, entre los que se destacan los siguientes:

Codificación Binaria: Es la más antigua de todas las existentes. La representación de los cromosomas, está definida como cadenas de bits, de modo que, dependiendo del problema, cada gen del cromosoma puede estar formado por una subcadena de varios bits.

Codificación Real: La codificación binaria presenta algunos inconvenientes cuando se trabaja con problemas que incluyen variables cuyos dominios son continuos: Excesiva longitud de los cromosomas, falta de precisión, etcétera. Una posible manera de evitar estos inconvenientes, es tomar un esquema de representación real. Cada variable del problema se asocia a un único gen que toma un valor dentro del intervalo especificado.

Codificación basada en orden: Este esquema está diseñado, específicamente, para problemas de optimización combinatoria, en los que las soluciones son permutación de un conjunto de los elementos determinados.

Estos ejemplos de posibles representaciones, nos dan una idea genérica del tipo de esquemas que se utilizan más comúnmente. Esto no implica que sean los únicos, o que no se puedan crear esquemas propios, que no tengan relación alguna con los comentados, si es que se adaptan mejor a un problema particular.
La mayoría de las veces, una codificación correcta es la clave de una buena resolución del problema. Generalmente, la regla heurística que se utiliza, es la llamada regla de los bloques de construcción, es decir, parámetros relacionados entre sí, deben de estar cercanos en el cromosoma. Por ejemplo, si queremos codificar los pesos de una red neuronal; una buena elección será, poner juntos todos los pesos que salgan de la misma neurona de la capa oculta.
En algunos casos, cuándo no se conoce de antemano el número de variables del problema, caben dos opciones: codificar también el número de variables, fijando un número máximo, o bien, lo cual es mucho más natural, crear un cromosoma que pueda variar de longitud. Para ello, claro está, se necesitan operadores genéticos que alteren la longitud. Normalmente, la codificación es estática, pero en casos de optimización numérica, el número de bits dedicados a codificar un parámetro puede variar, o incluso, lo que representen los bits dedicados a codificar cada parámetro, también puede variar.

5.2.2. Evaluación y Selección

Durante la evaluación, se decodifica el gen, convirtiéndose en una serie de parámetros de un problema, se halla la solución del problema a partir de esos parámetros, y se le da una puntuación a esa solución en función de lo cerca que esté de la mejor solución. A esta puntuación se le llama *fitness*.
Por ejemplo, supongamos que queremos hallar el máximo, de la función parábola invertida, con máximo en x=*1*. En este caso, el único parámetro del problema es la variable x. La optimización consiste en hallar un *x* tal que, *F(x)* sea máximo. Crearemos una población de cromosomas, cada uno de los cuales contiene una codificación binaria del parámetro *x*. Lo haremos de la forma siguiente: cada byte, cuyo valor está comprendido entre 0 y 255, se transformará para ajustarse al intervalo [-1,1], donde queremos hallar el máximo de la función.

Valor binario	Decodificación	Evaluación f(x)
10010100	21	0.9559
10010001	19	0.9639
101001	-86	0.2604
1000101	-58	0.6636

Tabla 5.1 - Ejemplo de evaluación y selección

El fitness determina siempre los cromosomas que se van a reproducir, y aquellos que se van a eliminar, pero hay varias formas de considerarlo para seleccionar la población de la siguiente generación:

- Usar el orden o rango, y hacer depender la probabilidad de permanencia, o evaluación de la posición en este.
- Aplicar una operación al fitness para escalarlo; como por ejemplo, el escalado sigma.
- En algunos casos, el fitness no es una sola cantidad, sino diversos números, que tienen diferente consideración. Basta con que tal fitness, forme un orden parcial, es decir, que se puedan comparar dos individuos; y decir, cuál de ellos es mejor. Esto suele suceder cuando se necesitan optimizar varios objetivos.

Una vez evaluado el fitness, se tiene que crear la nueva población, teniendo en cuenta que los buenos rasgos de los mejores se transmitan a esta. Para ello, hay que seleccionar a una serie de individuos encargados de la tarea. Esta selección, y la consiguiente reproducción, se puede realizar de tres formas principales:

Basado en el rango: en este esquema se mantiene un porcentaje de la población, generalmente la mayoría, para la siguiente generación. Se coloca toda la población por orden de fitness, y los M menos dignos son eliminados y sustituidos por la descendencia de alguno de los M mejores, con algún otro individuo de la población. A este esquema se le pueden aplicar otros criterios; por ejemplo, se crea la descendencia de uno de los paladines / amazonas, y esta sustituye al más parecido entre los perdedores. Esto se denomina Crowding.
Rueda de ruleta: se crea un pool genético, formado por cromosomas de la generación actual en una cantidad proporcional a su fitness. Si la proporción hace que un individuo domine la población, se le aplica alguna operación de escalado. Dentro de este pool, se seleccionan parejas aleatorias de cromosomas, y se emparejan, sin importar incluso que sean del mismo padre, (para eso están otros operadores, como la mutación). Hay otras variantes; por ejemplo, en la nueva generación se puede incluir el mejor representante de la generación actual. En este caso, se denomina método elitista.
Selección de torneo: se escogen, de manera aleatoria, un número T de individuos de la población, y el que tiene puntuación mayor se reproduce, sustituyendo su descendencia por la del de menor puntuación.

5.2.3. Operadores Genéticos

5.2.3.1. Crossover

Consiste en el intercambio de material genético entre dos cromosomas. El crossover (entrecruzamiento o recombinación) es el principal operador genético, hasta el punto que se puede decir que no es un algoritmo genético, si no tiene crossover; y, sin embargo, puede serlo perfectamente sin mutación.
Para aplicar el crossover, se escogen, aleatoriamente, dos miembros de la población. No pasa nada si se emparejan dos descendientes de los mismos padres; ello garantiza la perpetuación de un individuo con buena puntuación (y, además, algo parecido ocurre en la realidad; es una práctica utilizada, por ejemplo, en la cría de ganado, llamada inbreeding, y destinada a potenciar ciertas características frente a otras). Sin embargo, si esto sucede demasiado a menudo, puede crear problemas; toda la población puede aparecer dominada por los descendientes de algún gen, que además, puede tener caracteres no deseados. Esto se suele denominar en otros métodos de optimización, estancamiento en un mínimo local, y es uno de los principales problemas con los que se enfrentan quienes aplican algoritmos genéticos.
El intercambio genético se puede llevar a cabo de muchas formas, pero hay dos grupos principales
Crossover n-puntos: los dos cromosomas se cortan por n puntos, y el material genético situado entre ellos se intercambia. Lo más habitual es un crossover de un punto

Crossover uniforme: se genera un patrón aleatorio de 1s y 0s, y se intercambian los bits de los dos cromosomas que coincidan, donde hay un 1 en el patrón; o bien, se genera un número aleatorio para cada bit, y si supera una determinada probabilidad, se intercambia ese bit entre los dos cromosomas.

Crossover especializados: en algunos problemas, aplicar aleatoriamente el crossover da lugar a cromosomas que codifican soluciones inválidas; en este caso, hay que aplicar el crossover de forma que genere siempre soluciones válidas.

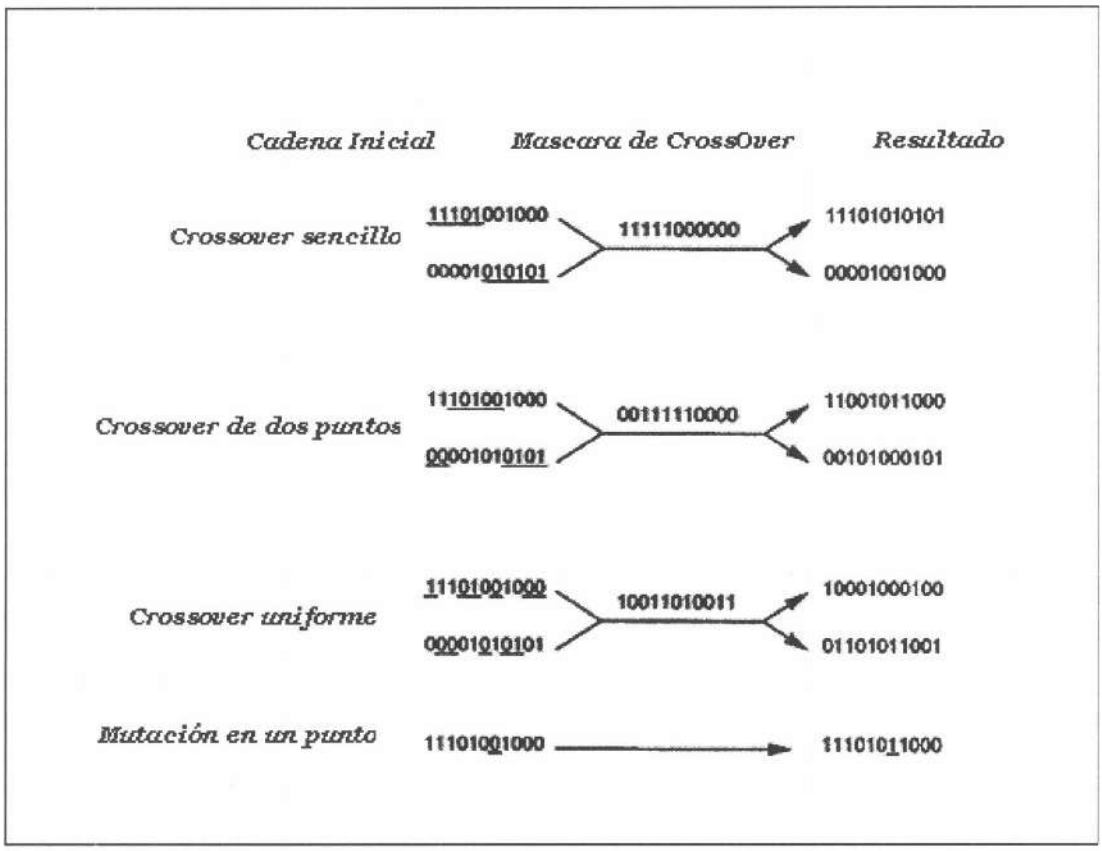

Figura 5.2 - Ejemplos crossover

5.2.3.2. Mutación

En la evolución, una mutación es un suceso bastante poco común, sucede aproximadamente una de cada mil replicaciones, en la mayoría de los casos las mutaciones son letales, pero en promedio, contribuyen a la diversidad genética de la especie. En un algoritmo genético tendrán el mismo papel, y la misma frecuencia, es decir, muy baja.
Una vez establecida la frecuencia de mutación, por ejemplo, uno por mil, se examina cada bit de cada cadena cuando se vaya a crear la nueva criatura, a partir de sus padres, normalmente se hace de forma simultánea al crossover. Si un número generado aleatoriamente está por debajo de esa probabilidad, se cambiará el bit (es decir, de 0 a 1 o de 1 a 0). Si no, se dejará como está. Dependiendo del número de individuos que haya y del número de bits por individuo, puede resultar que las mutaciones sean extremadamente raras en una sola generación.
No hace falta decir que no conviene abusar de la mutación. Es cierto que es un mecanismo generador de diversidad, y por lo tanto, la solución cuándo un algoritmo genético está estancado, pero también es cierto que reduce el algoritmo genético a una búsqueda aleatoria. Siempre es más conveniente usar otros mecanismos de generación de diversidad, como aumentar el tamaño de la población, o garantizar la aleatoriedad de la población inicial.

Este operador, junto con el anterior, y el método de selección de ruleta, constituyen un algoritmo genético simple, el mismo fue introducido por Goldberg [GOL/1989 II]

5.2.3.3. Otros Operadores

No se usan en todos los problemas, sólo en algunos, y en principio su variedad es infinita. Generalmente, son operadores que exploran el espacio de soluciones de una forma más ordenada, y que actúan más en las últimas fases de la búsqueda, en la cual se pasa de soluciones "casi buenas" a "buenas" soluciones.

Cromosomas de longitud variable

Hasta ahora se han descrito cromosomas de longitud fija, donde se conoce de antemano el número de parámetros de un problema. Pero hay problemas en los que esto no sucede. Por ejemplo, en un problema de clasificación, donde dado un vector de entrada, queremos agruparlo en una serie de clases, podemos no saber siquiera cuantas clases hay; o en diseño de redes neuronales, puede que no se sepa, de hecho, nunca se sabe, cuántas neuronas se van a necesitar. Por ejemplo, en un perceptrón hay reglas que dicen cuantas neuronas se deben de utilizar en la capa oculta; pero en un problema determinado, puede que no haya ninguna regla heurística aplicable; podremos utilizar los algoritmos genéticos para hallar el número óptimo de neuronas.
En este caso, necesitaremos un gen para cada neurona de la capa oculta, y el número de gen variará, dependiendo del número de neuronas de la capa oculta.
En estos casos, necesitamos dos operadores más: añadir y eliminar. Estos operadores se utilizan para añadir un gen, o eliminar un gen del cromosoma. La forma más habitual de añadir un gen, es duplicar uno ya existente, el cual sufre mutación, y se añade al lado del anterior. En este caso, los operadores del algoritmo genético simple (selección, mutación, crossover) funcionarán de la forma habitual, salvo, claro está, que sólo se hará crossover en la zona del cromosoma de menor longitud.
Estos operadores permiten además, crear un algoritmo genético de dos niveles: a nivel de cromosoma, y a nivel de gen. Supongamos que, en un problema de clasificación, hay un gen por clase. Se puede asignar una puntuación a cada gen en función del número de muestras que haya clasificado correctamente. Al aplicar estos operadores, se duplicarán los alelos con mayor puntuación, y se eliminarán aquellos que hayan obtenido menor puntuación, o cuya puntuación sea nula.

Operadores de nicho (ecológico)

Otros operadores importantes son los operadores de nicho. Estos operadores están encaminados a mantener la diversidad genética de la población, de forma que, cromosomas similares, sustituyan sólo a cromosomas similares; y son especialmente útiles en problemas con muchas soluciones. Un algoritmo genético con estos operadores es capaz de hallar todos los máximos, dedicándose cada especie a un máximo.
Más que operadores genéticos son formas de enfocar la selección y la evaluación de la población. Una de las formas de llevar esto a cabo es la introducción del crowding (apiñamiento). Otra forma es introducir una función de compartición o sharing, que indica, cuán similar es un cromosoma al resto de la población. La puntuación de cada individuo se dividirá por esta función de compartición, de forma que se facilita la diversidad genética, y la aparición de individuos diferentes.

También se pueden restringir los emparejamientos, por ejemplo, a aquellos cromosomas que sean similares; para evitar las malas consecuencias del inbreeding que suele aparecer en poblaciones pequeñas, estos periodos se intercalan con otros periodos en los cuales el emparejamiento es libre.

Operadores especializados

En una serie de problemas, hay que restringir las nuevas soluciones generadas por los operadores genéticos, pues no todas las soluciones generadas van a ser válidas, sobre todo en los problemas con restricciones. Por ello, se aplican operadores que mantengan la estructura del problema. Otros operadores son simplemente generadores de diversidad, pero la generan de una forma determinada:

- ***Zap***: en vez de cambiar un solo bit de un cromosoma, cambia un gen completo de un cromosoma.
- ***Creep***: este operador aumenta o disminuye en 1 el valor de un gen; sirve para cambiar suavemente y de forma controlada los valores de los genes.
- ***Transposición***: similar al crossover y a la recombinación genética, pero dentro de un solo cromosoma; dos genes intercambian sus valores, sin afectar al resto del cromosoma. Similar a éste es el operador de eliminación-reinserción, en el que un gen cambia de posición con respecto a los demás.

5.2.3.4. Aplicando Operadores Genéticos

En toda ejecución de un algoritmo genético, hay que decidir con qué frecuencia se va a aplicar cada uno de los operadores, en algunos casos, como en la mutación, o el crossover uniforme, se debe de añadir algún parámetro adicional, que indique con qué frecuencia se va a aplicar dentro de cada gen del cromosoma. La frecuencia de aplicación de cada operador estará en función del problema. Teniendo en cuenta los efectos de cada operador, tendrá que aplicarse con cierta frecuencia o no. Generalmente, la mutación y otros operadores que generen diversidad, se suelen aplicar con poca frecuencia; la recombinación se suele aplicar con frecuencia alta.
En general, la frecuencia de los operadores no varía durante la ejecución del algoritmo, pero hay que tener en cuenta, que cada operador es más efectivo en un momento de la ejecución. Por ejemplo, al principio, en la fase denominada de exploración, los más eficaces son la mutación y la recombinación; posteriormente, cuándo la población ha convergido en parte, la recombinación no es útil, pues se está trabajando con individuos bastante similares, y es poca la información que se intercambia. Sin embargo, si se produce un estancamiento, la mutación tampoco es útil, pues está reduciendo el algoritmo genético a una búsqueda aleatoria; y hay que aplicar otros operadores. En todo caso, se podrían usar operadores especializados.

5.3. Propiedades de los Algoritmos Genéticos

5.3.1. Lenguaje de Representación

El aprendizaje de los algoritmos genéticos está basado en la simulación de la evolución, en cualquiera de sus variantes, es decir, por mutación o por selección natural. Las hipótesis que se aprenden originalmente son representadas por medio de cadenas de bits, cuya interpretación depende del tipo de la aplicación y la función objetivo.

5.3.2. Sesgo

5.3.2.1. Sesgo Inductivo

El sesgo inductivo viene dado por la codificación de los genes, los cromosomas, la población, la función de fitness, y porcentaje de cross-over, porcentaje de mutación, etcétera; es decir, cada uno de los parámetros de los algoritmos genéticos produce un sesgo, siendo critica la tarea de codificación.

5.3.2.2. Sesgo de la Búsqueda:

La búsqueda de una hipótesis apropiada comienza con una población, o colección, de hipótesis iniciales. Los miembros de la actual población pasan a la próxima generación, por medio de operaciones tales como mutación o cruce (es aquí donde se introduce un fuerte sesgo), asociadas a procesos de evolución biológica. En cada paso, la hipótesis es evaluada sobre la base de una medida de aptitud, y las mejores son seleccionadas en forma probabilística para pasar a la próxima generación.
En principio, se ven a los algoritmos genéticos como métodos de búsqùeda independientes del dominio.
Sin embargo puede incorporarse conocimiento del dominio en:

- los operadores genéticos a utilizar
- la representación usada
- poblaciones iniciales
- retroalimentación

5.3.3. Tipos de Problemas Apropiados

- Problemas de reconocimiento de patrones.
- Lectura – Escritura Sintaxis de un lenguaje
- Problemas de Clustering.
- Problemas de Búsqueda compleja.

5.3.4. Comportamientos Comunes:

Algunos problemas que surgen de la utilización de los algoritmos genéticos son:

Efecto Baldwin: Baldwin (1896) propuso que si existen cambios en el ambiente, la evolución tiende a favorecer individuos con la capacidad de aprender a adaptarse al nuevo medio ambiente, acelerando cambios genéticos entre individuos parecidos.
Crowding: es el fenómeno ocurrido cuando algún individuo es mucho más apto que los demás, y se reproduce rápidamente llenando la población con sus copias. Esto reduce diversidad, y desacelera rápidamente el proceso evolutivo.

5.4. Algoritmos Genéticos Para Generar Perfiles Dinámicos de Usuarios

Debido a las consideraciones tenidas en cuenta en las secciones anteriores, podemos decir que utilizar algoritmos genéticos para generar perfiles dinámicos, es una tarea digna de investigación. Para esto se define una población de individuos (perfiles en nuestro caso) que luchan por obtener la mejor calificación de la función de fitness. Esto es en general; en particular tenemos que definir, cómo serán los individuos, cuántos cromosomas tendrán, cómo se relacionan los cromosomas, y los atributos del dominio, etcétera. Si alguno de estos conceptos no le resulta claro es recomendable remitirse a la sección 5.1.

5.4.1. Representación de Perfiles Utilizando Algoritmos Genéticos

Dijimos que podemos abordar el problema de la generación de perfiles dinámicos de usuarios, a través de algoritmos genéticos. En este enfoque se cuenta con una población de individuos, cada uno representando un perfil. A continuación daremos un ejemplo sobre como podemos llevar esto a la práctica; pero, antes de ello, es válido aclarar que el perfil, propiamente dicho, del usuario, está formado, en el caso de algoritmos genéticos, por el individuo de mejor calificación de la población. Cada individuo representa un perfil. Luego el algoritmo utilizará el mejor perfil (individuo) cada vez que se requiera un diagnóstico sobre un determinado caso.

5.4.1.1. Representación

A continuación damos una posible configuración de un algoritmo genético para resolver nuestro problema. Ésta es sólo una configuración de ejemplo, utilizada para: Primero, introducir la notación que utilizaremos en la representación de los casos analizados. Segundo, fijar los conceptos vertidos hasta aquí sobre algoritmos genéticos, y su uso para representar perfiles dinámicos de usuarios.
Supongamos que tenemos la siguiente representación, cada cromosoma representa un cierta cantidad variable de valores de atributos. Cada valor de un atributo se corresponde con un gen de algún determinado cromosoma, y supongamos, que cada individuo posee 6 cromosomas. De esta manera un individuo estaría representado, gráficamente, de la siguiente manera.

Figura 5.3 – representación de un Individuo

En este caso, la *Figura* 5.3, representa un individuo, cada rectángulo interno, representa un cromosoma, y cada línea dentro del cromosoma un gen. En cada gen, y en negrita se aprecia el nombre del atributo, y separado por un carácter dos puntos, el valor del atributo.
Esta representación nos da una idea gráfica bastante clara, de cómo está formado el individuo, y cada uno de los componentes; pero, para poder utilizar un lenguaje de representación más formal y compacto, introduciremos la siguiente notación, la cual será utilizada más adelante, en la especificación de las distintas configuraciones tenidas en cuenta.
Sea:

I_i el i-ésimo Individuo de la población.

C_i el i-ésimo cromosoma correspondiente al algún individuo.

C_{ji} el j-ésimo cromosoma correspondiente al i-ésimo individuo.

G_j el j-ésimo gen perteneciente a algún Cromosoma.

G_{ji} el j-ésimo gen perteneciente al cromosoma i-ésimo.

G_{jik} el j-ésimo gen perteneciente al cromosoma i-ésimo del k-ésimo individuo

A_i el i-ésimo de los atributos definidos en la *Tabla 5.3.*

$A_{i(j)}$ el j–ésimo valor del atributo A_i. Por ejemplo si A_i= *Target y target= {mujer, niño, hombre, adolescente, unisex}* entonces $A_{i(3)}$ Es igual a hombre.

#A la cantidad total de Atributos.

#A_i la cantidad de elementos del atributo A_i, en el caso de *target,* es igual a 5.

#C_i la cantidad de genes del cromosoma i-ésimo.

#I_i la cantidad de cromosomas del i-ésimo individuo

#I la cantidad de individuos en la población

Utilizando esta notación, podemos representar al individuo de la *Figura* 5.3, de la siguiente manera.

$$I_1 = \{ C_1, C_2, C_3, C_4, C_5, C_6 \}$$

Si por ejemplo tomamos el cromosoma numero 4 del individuo, contando los cromosomas de izquierda a derecha, y de arriba hacia abajo, según el gráfico. Esta posición, corresponde al cromosoma que contiene solamente un gen; y que representa al atributo *Talle.* Éste será representado de la siguiente manera.

$$C_4 = \{ G_1 \}$$

A su vez el gen se representa de la siguiente manera.

$$G_1 = \{ A_{16}, 1 \}$$

Donde

$$A_{16} = \textit{Talle} \text{ y } A_{16,1} = \text{XL } (\textit{eXtra Large})$$

Representan el nombre del atributo y el valor del atributo, respectivamente.

Ahora explicaremos las distintas acciones y funciones del algoritmo genético. Lo haremos de ambas maneras, gráfica, y formalmente, a través de la notación introducida.

5.4.1.2. Crossover

En el gráfico de la *Figura* 5.4 se muestra cómo sería un crossover entre dos cromosomas de dos individuos distintos. El crossover puede realizarse en cada uno, o entre todos los cromosomas del individuo; también puede hacerse parcialmente, seleccionando algunos cromosomas, según algún parámetro. En nuestro caso, para este ejemplo, optamos por un crossover completo, es decir entre todos los cromosomas.

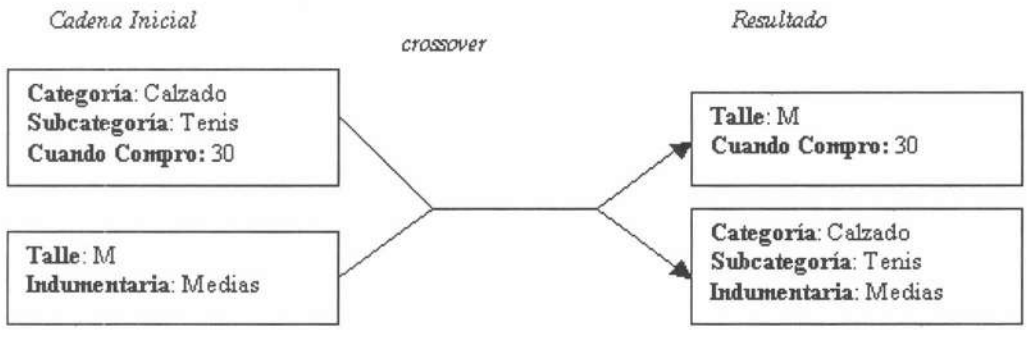

Figura 5.4 – Ejemplo Crossover

En la *Figura* 5.4 puede verse cómo se generan dos nuevos individuos, y cómo estos heredan parte de los atributos de cada padre. Utilizando la codificación presentada anteriormente, estos se representarían de la siguiente manera:

Sean I_p e I_k, el primero y el segundo de los padres, con los que se realizarán el crossover, respectivamente. $C_{i,p}$ es el i-ésimo cromosoma del Individuo p y $C_{i,k}$ es el i-ésimo cromosoma del individuo k.

El crossover entre estos se define de la siguiente manera

Sea

$$h = \min(\#C_{i,p}, \#C_{i,k})$$

Donde h es un valor entero conteniendo la cantidad de genes del individuo con menos genes.

$$j = \text{random}(1, h)$$

Donde j un valor entero elegido al azar, comprendido entre uno y la cantidad de genes del individuo con menos genes.

Sean $C'_{i,p}$, $C'_{i,k}$ los hijos de $C_{i,p}$, $C_{i,k}$, surgidos luego del crossover y sea $\oplus$ el símbolo utilizado para representar el crossover. Entonces.

$$C_{i,p} \oplus C_{i,k} = C'_{i,p}, C'_{i,k}$$

Con

$$C'_{i,k} = \{i \mid 1 \le i < j, G_{i,k}\} \cup \{i \mid j \le i \le \#C_{i,p}, G_{i,p}\}$$

$$C'_{i,p} = \{i \mid 1 \le i < j, G_{i,p}\} \cup \{i \mid j \le i \le \#C_{i,k}, G_{i,k}\}$$

5.4.1.3. Mutación

Otro de los operadores genéticos importantes es la mutación, que en nuestro caso se da a nivel de gen (Zap); es decir, se cambia un determinado gen de manera completa, sustituyéndolo por otro.

En el ejemplo de la *Figura 5.5* se puede ver cómo cambian los valores de los genes correspondientes al atributo *Categoría,* cuyo valor es *Calzado,* y el atributo *CuandoCompro* cuyo valor es 30; estos son cambiados por los valores, indumentaria del atributo *Categoría,* y por el valor *camisetas,* también perteneciente al atributo *Indumentaria,* respectivamente.

Figura 5.5 – Ejemplo de mutación

Este ejemplo, muestra también, que no solamente, un solo gen puede ser mutado a la vez, por el contrario, pueden ser varios genes de un mismo cromosoma los que se vean afectados, dependiendo de la probabilidad de mutación existente en la población. Utilizando nuestra representación, tenemos que una mutación ocurre de la siguiente manera.
Sea

$$I_k^t$$

un individuo cualquiera I_k de la población, al que se le ha aplicado el operador de mutación, notado con el superíndice *T*.
Sea:

$$\text{donde } n = \text{random}(1, \#A)$$

n es un entero, elegido al azar, entre 1 y la cantidad de atributos del dominio.

$$G_{jik} = \begin{cases} \{A_n, \text{random}(1, \#A_n)\} & \text{Si random}(0,1) > 0 \\ G_{jik} & \text{Si random}(0,1) = 0 \end{cases}$$

Es decir, dado un gen cualquiera, en este caso señalado como el j-ésimo gen perteneciente al cromosoma i-ésimo, del individuo k, tendrá un nuevo valor con una probabilidad del 50 %, es decir la posibilidad de que se elija al azar (random) un entero entre 0 y 1; y, ese nuevo valor, será un atributo elegido al azar entre todos los atributos (random(1, #A)), y el valor seleccionado del atributo, también es escogido al azar (random(1, #An)).

5.4.1.4. Nuevos Genes

Otro operador que utilizamos, aunque no es muy común, es el que permite agregar nuevos genes a un cromosoma, por supuesto que la cantidad mínima de genes de un individuo, deberá estar determinada por algún parámetro. En la *Figura* 5.6 puede observarse un ejemplo.

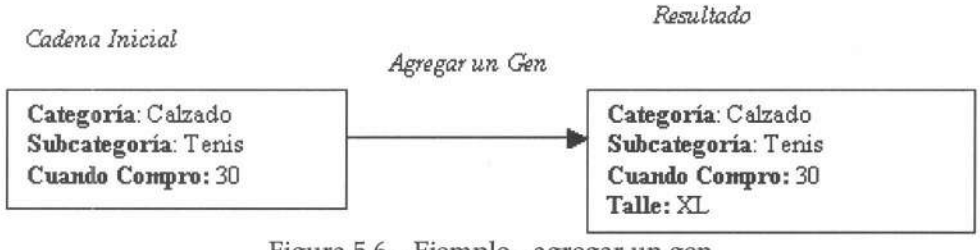

Figura 5.6 – Ejemplo - agregar un gen

Como podemos observar luego de aplicarle la operación de agregar un gen, al cromosoma, este cuenta con un atributo más, el *Talle*.
Este se aplica al igual que en el ejemplo anterior, de manera aleatoria a cada cromosoma, formalmente sería.

Sea

$$I_i^+$$

Un individuo cualquiera, I_i de la población, al que se le ha aplicado el operador de mutación, notado con el superíndice +.

$$\text{donde } n = \text{random}(1, \#A)$$

n es un entero, elegido al azar, entre 1 y la cantidad de atributos del dominio.

$$C_{ji} = \begin{Bmatrix} \{k | G_{kji}\} \cup \{A_n, \text{random}(1, \#A_n) & \text{Si random } (0,1)>0\} \\ C_{ji} & \text{Si random } (0,1)=0 \end{Bmatrix}$$

La traducción de esta formula sería: en el 50 % de los casos agregamos un gen a los cromosomas, en la primer línea de la formula se expresa, que el cromosoma será igual a todos los genes que tenía, unidos al nuevo gen, formado este, por un atributo elegido al azar.

5.4.1.5. Fitness

Como vimos en la primera parte de este capítulo, un algoritmo genético no es nada si no se puede calcular el fitness de sus individuos. Mostraremos ahora una manera sencilla de calcular el fitness para nuestro ejemplo. Supongamos la siguiente función. Para cada uno de los atributos que conforman el caso, sumaremos 1 al fitness, si un cromosoma contiene el valor de un atributo que figura en el caso a diagnosticar. El valor dado por todos los cromosomas es sumado, y ese será el fitness del individuo.
Como ejemplo podemos tomar el caso de la *Figura 5.7* y el individuo representado en la *Figura* 5.3.

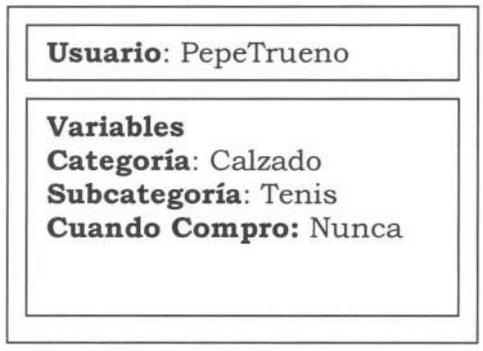

Figura 5.7 - Caso con el que se medirá el fitness de un individuo

En este caso los atributos presentes en el caso son:

Atributos = {Categoría=Calzado, Subcategoría=Tenis, CuandoCompro=Nunca}

Y el valor que nos daría la suma para cada cromosoma, contando de izquierda a derecha, y de arriba hacia abajo, para cada cromosoma sería.

$$C_1= 2,\ C_2= 0,\ C_3= 0,\ C_4=0,\ C_5= 1,\ C_6= 1$$

Y el fitness de este individuo sería entonces de 4, que es la suma de los valores de los cromosomas.
De manera formal, utilizando nuestra representación: introduciremos el nuevo símbolo eval, el cual queda definido de la siguiente manera.

$$\mathrm{Eval}(G_{ijk}) = 1 \text{ si } G_{ijk} \in \text{Atributos}$$

$$\mathrm{Eval}(C_{jk}) = \sum_{i=1}^{\#C_{jk}} Eval(G_{ijk})$$

$$\mathrm{Eval}(I_k) = \sum_{j=1}^{\#C_k} Eval(C_{jk})$$

Con lo que concluimos que

$$\text{Fitness } (I_k) = \text{Eval } (I_k)$$

Es decir que Eval(I) representa el fitness del individuo I.

5.4.1.6. Nueva Generación

Para determinar que individuos pasan a la siguiente generación, habrá que evaluar los parámetros señalados anteriormente, y la función de fitness para cada individuo; una vez realizado esto, pasarán sin cambio a la siguiente generación el 5% de los individuos más aptos, seleccionaremos y cruzaremos el 30% de los individuos más aptos restantes, mutaremos aleatoriamente 50% de los restantes, y al otro 50 % restante se le agregarán genes y restarán genes.

.

5.4.1.7. Perfil

De esta manera, es como cada individuo representa un perfil; y cómo podemos representarlo utilizando algoritmos genéticos.
En los párrafos siguientes se explica cuáles fueron las configuraciones, utilizadas en la implementación de la técnica de aprendizaje algoritmos genéticos, para la construcción de nuestro agente de comercio electrónico: *Vendor*.

5.5. Generando Perfiles Dinámicos de Usuarios Utilizando Algoritmos Genéticos

A continuación se detallan una serie de configuraciones e implementaciones de algoritmos genéticos, utilizados para resolver el problema de aprender un perfil dinámico de usuario, para tiendas de comercio virtual. Estas implementaciones fueron surgiendo a lo largo del análisis de la técnica de aprendizaje genético para nuestro agente. Creemos que es fundamental transcribir las implementaciones más importantes realizadas. A beneficio de quienes tengan que implementar un algoritmo de este tipo.
Para la implementación del algoritmo genético se recurrió a un framework ya existente [GRO/2000], su elección se debe a que es uno de los más configurable, adaptable, y extensible de los analizados. Éste permite extender una clase mediante la cual se tiene implementado el corazón del algoritmo (Ver Figura 5.1).
Quien quiera utilizar este framework deberá extender esta clase, sobrescribir los métodos que desee modificar, y configurar cada uno de los parámetros de un algoritmo genético (Cromosomas, Individuos, Genes etcétera). Cabe destacar que este framework fue reimplementado en parte, debido a que dotamos, como más adelante se detalla, de memoria a los individuos para poder recordar la importancia de los atributos.

5.5.1. Parámetros del Algoritmo

A continuación se detalla cada uno de los parámetros a configurar para la ejecución correcta del algoritmo, que por supuesto, incluye los parámetros de un algoritmo genético estándar.
Es válido decir que sólo daremos una explicación superficial, para poder entender las implementaciones realizadas. Estos parámetros son los que se describen en la siguiente sección, y son comunes a cualquier algoritmo genético. Si se desea más información sobre el framework es conveniente referirse a la bibliografía[GRO/2000]

5.5.1.1. World (Mundo)

En el Mundo se define, la función de Fitness, la cantidad de Individuos, el porcentaje de crossover, el porcentaje de mutación, etcétera. El mundo incluye a todos lo elementos del algoritmo genético

5.5.1.2. Genes

Los genes, en el framework, se especifican como un elemento perteneciente a un conjunto formado por Variables y Funciones. Donde las variables son parámetros que pueden modificarse en cada iteración del algoritmo, las variables corresponderían a parámetros recibidos por el algoritmo, y las funciones predefinidas por quien configura el framework. El concepto de Variable y Función es amplio y abarca funciones multidimencionales y objetos (Array y Objects). Dando un amplio poder de representación. Por ejemplo, En nuestro caso, y utilizando la configuración de la sección anterior, podemos utilizar variables para representar cada atributo.

5.5.1.3. Cromosomas

Los cromosomas, básicamente, son un conjunto de Genes. La longitud de los cromosomas puede ser fija o variable. En nuestro caso, usamos cromosomas variables en todas las configuraciones. Por ejemplo, podemos tener La Función 1 (constante uno) y la Función + (suma), los cuales son a su vez genes. Entonces, un posible cromosoma sería 1 + 1. De esta manera, queda claro que los genes se agregan al azar; pero respetando un cierto criterio. Por ejemplo: el Cromosoma, 1 1 +, carece de sentido y no es un cromosoma válido.
Este tipo de configuraciones es útil, y sirve para tener una representación clara del dominio en la especificación de los parámetros del algoritmo, a su vez, agregar sólo aquellos métodos que generan cromosomas válidos, hacen que la técnica sea más robusta. Las funciones no necesariamente son de tipo entero pueden ser matrices, Strings, booleanos, o algún otro tipo de función implementada por nosotros. La longitud máxima de un cromosoma es configurable.

5.5.1.4. Individuos

Los individuos son un conjunto de cromosomas, y su memoria, en los casos en que utilizamos este concepto.

5.5.1.5. *Otros Parámetros*

Parámetro	Descripción
Método de Selección	Tipo de selección de individuos. Ranking, Azar, etcétera
Tipo de Cross over	Múltiple. Simple. Etcétera
Probabilidad de Cross Over	Porcentaje de individuos a los que se les realizará cross – over
Cantidad Máxima de Genes	Cantidad de genes máxima en un cromosoma
Cantidad Mínima de Genes	Cantidad de genes mínimas en un cromosoma.
Probabilidad de Mutación	Probabilidad de que se realice una mutación
Individuos Mantenidos	Cantidad de individuos que se mantienen en la próxima generación
Probabilidad gen nuevo	Probabilidad de agregar un gen a un cromosoma

Tabla 5.2 - Otros Parámetros del algoritmo genético

5.6. Aprender Perfiles por Medio de Algoritmos Genéticos, Enfoques Posibles Para Resolver Este Problema

Para poder entender las configuraciones, primero, explicaremos cómo se codifican los casos, y cómo se tratan los distintos atributos, según sean, discretos o continuos.
Como ya sabemos, un caso tiene asociados, según cuál sea la operación que este realizando, una cierta cantidad de atributos. Estos atributos son un elemento del conjunto, formado por las partes del conjunto de atributos.
Estos serán tratados de la siguiente manera.

Nombre	Descripción	tipo dominio	Tratamiento
Categorías	Categoría del producto	Discreto	Discreto
Accesorios	Categoría accesorios	Discreto	Discreto
Calzado	Categoría calzados	Discreto	Discreto
Deportes	categoría deportes	Discreto	Discreto
Entrenamiento	categoría entrenamiento	Discreto	Discreto
Indumentaria	categoría indumentaria	Discreto	Discreto
Marcas	categoría marcas	Discreto	Discreto
CategoríaComprador	a quién está destinado el producto	Discreto	Discreto
Catprecio	categoría del precio	Discreto	Discreto
Exposición	tiempo de exposición	Continuo	Discreto
Fecha	Fecha de Hoy	Continuo	Discreto
Color	Colores	Discreto	Discreto
RankingInterno	Categorización del desempeño del producto dentro de la empresa	Discreto	Discreto
CuandoCompro	tiempo transcurrido desde la compra	Continuo	Discreto
Calidad	Calidad del producto	Discreto	Discreto
Talle	Talle del producto	Discreto	Discreto
Interés	Concepto objetivo interés dado el producto	Continuo	Continuo

Tabla 5.3 - Variables del dominio

Todos los atributos continuos se traducen de la misma manera que se realizó en la implementación de la red bayesiana.
Esto se debe principalmente a que de otra manera los algoritmos no demostraron una buena performance. Principalmente, debido a la fecha, ya que esta medida en días es muy amplia, y dividida en días, semanas, y meses, es más compacto, y, por ende, más eficiente.

5.6.1. Implementación I

A esta configuración podríamos llamarla ingenua, debido a que se basa en una clásica y sencilla receta de los libros. En ésta se mapea cada elemento de cada atributo al valor cero, o a al valor uno, siendo cero si el atributo no está presente, y uno en caso contrario.

5.6.1.1. Individuos

Los individuos constan de un cromosoma de largo variable. Formado por una cadena de genes perteneciente al conjunto formado por la unión de los valores de los atributos, más cuatro genes representando funciones matemáticas. Las cadenas se forman seleccionando genes al azar. Siempre y cuándo formen una cadena compatible. Esto se verá más adelante.

5.6.1.2. Cromosoma

El cromosoma puede estar formado por una cadena de genes, los cuales, pertenecen al conjunto formado por la unión de Cuatro genes que representan Funciones matemáticas (+, -, *, /). Más la representación binaria de los atributos, más tres funciones constantes representando los valores 1,-1, y 0.
Es decir, de manera formal, existe F biyectiva tal que

$$\forall(i,j)(0 \le i <\# A \wedge 0 \le j <\# Ai) \Rightarrow \exists(k)(k \le (\sum_{h=0}^{\#A} \# A_h) \wedge F(Ai_{(j)}) = G_k)$$

Esto quiere decir que, G_k es un gen que representa a un elemento de algún atributo; y que, a cada valor perteneciente a algún atributo, le corresponde uno y sólo un G_k.

Al evaluar G_k en una función de fitness, pasa que: $eval(G_k) = 0$, si este valor no participa en el caso que se está evaluando; o bien $eval(G_k) = 1$, si es un valor de un atributo del caso que se está evaluando en este momento.

Además Existen los siguientes genes

G_+ tal que representa a la función +.
G_- tal que representa a la función –
$G_/$ tal que representa a la función /
G_* tal que representa a la función *.
G_1 tal que representa a la función constante 1.
G_{-1} tal que representa a la función constante -1.
G_0 tal que representa a la función constante 0.

De esta manera el cromosoma podría estar formado, por ejemplo, los siguientes genes.

$$C_1 = G_1\ G_*\ G_{25}\ G_+\ G_2\ G_+\ G_3\ G_-\ G_9\ G_*\ G_{25}\ G_+\ G_1\ G_+\ G_1\ G_+\ G_0$$

Para ver cuál es la representación de este cromosoma y cómo se evalúa un caso supongamos que $G_1 = 1$ y $G_2=1$ y $G_x = 0$ (Para todo x>1). G1 y G2 son los únicos elementos que participan en el caso (por ejemplo oferta perteneciente al atributo precio, y Nike perteneciente al atributo Marca)

$$C_1 = G_1\ G_*\ G_{25}\ G_+\ G_2\ G_+\ G_3\ G_-\ G_9\ G_*\ G_{25}\ G_+\ G_1\ G_+\ G_1\ G_+\ G_0$$
$$\Rightarrow Eval(C_1) = 1 * 0 + 1 + 0 - 0 * 0 + 1 + 1 + 0$$
$$\Rightarrow Eval(C_1) = 1 + 1 + 1 = 3$$

El cromosoma 1 le da un valor de 3 al caso que posee los atributos:

Atributos = {Marca= Nike , Precio= Oferta}

5.6.1.3. Fitness

La función de fitness retornará un valor alto para aquellos atributos que se creen mejor para el usuario. Nos quedamos con aquellos individuos que más valor le den al caso actual. Además, en caso que el usuario compre un producto no sugerido se aprenderán los atributos de este producto, seleccionando aquellos individuos que más valor le den a estos atributos.

5.6.1.4. Próxima Generación

Los demás parámetros de configuración utilizados fueron:

Parámetro	Descripción
Método de Selección	Se seleccionan los primeros individuos más aptos
Tipo de Cross over	Croosover multipunto a nivel de gen.
Probabilidad de Cross Over	60%
Cantidad Máxima de Genes	500
Cantidad Mínima de Genes	20
Probabilidad de reproducción	20%
Individuos Mantenidos	10%
Probabilidad gen nuevo	10%

5.6.1.5. Conclusiones

Creemos, que esta representación es demasiado ingenua, pudiendo servir solamente para dominios muy sencillos, y sobre todo para generar diversidad. De hecho, existen trabajados en los que se han utilizado los algoritmos genéticos como fuente de diversidad para la generación de perfiles de usuario[BEE/1994], utilizando configuraciones sencillas. Otra opción para elegir la configuración, es el tiempo de implementación, relativamente corto.

5.6.2. Implementación II

En este caso intentamos evolucionar a partir de la configuración anterior; y permitir que no sólo sean funciones matemáticas, sino reglas, del tipo *if –then –else*, de este modo le damos otro tipo expresividad a la formula. Además cambiamos la representación de los atributos, ahora tenemos un gen por cada atributo, y podemos preguntar a que elemento del conjunto representa. Por ejemplo: si tenemos el atributo:

precio = {Oferta, Liquidación, Normal}

Tendremos un Gen precio, y podremos preguntar sí Precio = Oferta. O lo que es lo mismo, si Precio =1 que es el índice de oferta dentro del atributo precio

5.6.2.1. Individuos

Los individuos están formados por un cromosoma con cadena de genes variables.

5.6.2.2. Cromosoma

El cromosoma está formado por un gen para cada atributo, más ciertos genes que representan funciones booleanas y aritméticas. Así por ejemplo un posible cromosoma podría representar la siguiente expresión.

*If Categoria==categoria_calzado then 1+1+1+1 else 0 + if precio=precio_oferta then 1+1 * if comprador==comprador_hombre then 1+1+0-1 else ...*

Donde *Precio* es un determinado valor perteneciente a los precios posibles y oferta es una constante definida que representa el precio oferta.

De esta manera logramos obtener cromosomas muchos más cortos que los anteriores y representar reglas de una manera bastante sencilla.

Formalmente tenemos la siguiente representación:
Sea F biyectiva tal que

$$\forall(i,j)(0 \leq i <\# A \wedge 0 \leq j <\# Ai) \Rightarrow F(Ai_{(j)}) = G_i)$$

(Para cada atributo Ai un Gen Gi)

Además existen los siguientes genes.

G = Gen que representa la igualdad entre dos valores incluyendo la igualdad entre un atributo y su valor (Precio = Oferta)

Gite Gen que representa el if then else

G+ tal que representa a la función +.

G- tal que representa a la función –

G/ tal que representa a la función /

G* tal que representa a la función *.

G1 tal que representa a la función constante 1.

G-1 tal que representa a la función constante -1.

G0 tal que representa a la función constante 0.

Luego tenemos genes Booleanos que reciben tres parámetros, el primero debe devolver un valor booleano, y los otros un valor real. Por eso agregamos como un gen más, la función de igualdad que devuelve un booleano. Además de valores enteros constantes para representar los valores de los atributos.

5.6.2.3. Fitness

La función de fitness sigue los mismos criterios que para la Configuración I.

5.6.2.4. Próxima Generación

Parámetro	Descripción
Método de Selección	Se seleccionan los primeros individuos más aptos
Tipo de Cross over	Croosover multipunto a nivel de gen.
Probabilidad de Cross Over	50%
Cantidad Máxima de Genes	300
Cantidad Mínima de Genes	20
Probabilidad de reproducción	30%
Individuos Mantenidos	10%
Probabilidad gen nuevo	10%

5.6.2.5. Conclusiones

Si bien esta configuración obtuvo resultados satisfactorios, esto se llevó a cabo con largos tiempos de proceso. Por otro lado, los cromosomas están representando reglas, y creemos que si el dominio puede expresarse de esta manera sería mejor utilizar algún tipo de algoritmos basado en reglas, o lógica difusa.
También vimos que al aplicar esta configuración del algoritmo, reglas básicas no son aprendidas; por ejemplo, el caso en que una persona compra un par de zapatillas, entonces inmediatamente el agente ofrece zapatillas similares, esto es un claro error que esta configuración no pudo sortear exitosamente, y que hace pensar en técnicas de aprendizaje híbridas para resolver este tipo de problemas.

5.6.3. Implementación III y la Utilización de Memoria

Los malos desempeños de las configuraciones anteriores hacen que sea necesario un replanteo del problema; si no, la modificación del algoritmo básico.
Considerando que un algoritmo genético, trata de imitar o tomar ideas de la evolución de las especies para llevarlas a la solución de un problema. Tenemos que hay un hecho importante en la evolución del hombre que hace a su supervivencia y desempeño (fitness); esta es la memoria. Por eso es que dotamos al framework que utilizamos de una memoria para los individuos. Ésta nos permite guardar recuerdos (funciones o valores) a los cuales se les asocia un valor de importancia, de esta manera con las constantes evoluciones, un determinado recuerdo puede ir cambiando su valor en la memoria y por ende modificando su consideración en el tiempo.
Para poder modificar la memoria se agregan a los cromosomas unos Genes especiales que implementan las distintas funcionalidades de la memoria. (mejorarRecuerdo, empeorarRecuerdo, recordar)
La noción de memoria para algoritmos genéticos ya a sido utilizada en [GME/2001], [GAN/2002] y [SUS/1997].

5.6.3.1. Individuos

Los individuos cuentan con un solo cromosoma el cual contiene información sobre todo los atributos, y su recuerdo es modificado durante la evolución.

5.6.3.2. *Cromosoma*

Un solo cromosoma conteniendo la información sobre todos los atributos, codificados como en el caso uno; además se agregan tres Genes para la utilización de la memoria del individuo estos son Recordar, Mejorar Recuerdo, Empeorar Recuerdo. Los representáremos por Gr, Gmr, y Ger respectivamente.

De manera formal, podemos representarlos de la siguiente manera:
Sea F biyectiva tal que:

$$\forall(i,j)(0 \le i <\# A \wedge 0 \le j <\# Ai) \Rightarrow \exists(k)(k \le (\sum_{h=0}^{\#A} \# A_h) \wedge F(Ai_{(j)}) = G_k)$$

(O sea que hay un gen para cada valor, de cada posible atributo, más los siguientes genes.)

G_+ tal que representa a la función +.

G_- tal que representa a la función –

$G_/$ tal que representa a la función /

G_* tal que representa a la función *.

G_1 tal que representa a la función constante 1.

G_{-1} tal que representa a la función constante -1.

G_0 tal que representa a la función constante 0.

G_r Tal que representa un Recuerdo de un atributo

G_{mr} Tal que representa la mejora de un recuerdo

G_{er} Tal que representa la disminución de importancia de un recuerdo.

Sea un i un valor entero
G_r es tal que

$$\forall(k)(0 \le k \le \sum_{h=0}^{\#A} \# A_h \Rightarrow \exists i \mid (G_r G_k) = i)$$

Es decir devuelve un valor i para cada elemento de cada atributo posible.

5.6.3.3. *Fitness*

La función de fitness sigue los mismos criterios que para la Configuración I

5.6.3.4. Próxima Generación

Parámetro	Descripción
Método de Selección	Se seleccionan los primeros individuos más aptos
Tipo de Cross over	Croosover multipunto a nivel de gen.
Probabilidad de Cross Over	70%
Cantidad Máxima de Genes	500
Cantidad Mínima de Genes	20
Probabilidad de reproducción	10%
Individuos Mantenidos	10%
Probabilidad gen nuevo	10%

5.6.3.5. Conclusión

La implementación de la memoria no resultó una mala idea; pero los tiempos de aprendizajes se hacen aun mayores debido principalmente a la configuración de lo cromosoma.

5.6.4. Implementación IV

Hemos variado un poco la implementación. Principalmente en la cantidad de cromosomas, y haciendo que estos tengan una labor específica. De esta manera acortamos el tamaño de los cromosomas.

5.6.4.1. Individuos

La cantidad de cromosomas de los individuos es igual a la cantidad de atributos más uno. Este último trata de agrupar a todos los atributos en una única función. Donde hay un gen por atributo, y el cromosoma representa por ejemplo:

*(1+1+1) * Marca + Precio + 2*

5.6.4.2. Cromosoma

Un cromosoma por cada uno de los atributos, cada uno de esto en cada iteración cambia el recuerdo sobre el atributo dándole un mayor o menor valor. Más un cromosoma que agrupa mediante operaciones matemáticas la importancia relativa de los atributos.

5.6.4.3. Fitness

La función de fitness se calcula mediante la suma la importancia de cada atributo del caso a evaluar por una constante K, Sumado a la evaluación del último cromosoma por una constante J.
Las constantes K, y J fueron dos grados de libertad que manejamos en la función de fitness para darle mayor o menor importancia al último cromosoma. Que, por ser más largo, daba en general valores muy grandes que hacían despreciable la memoria. El valor ideal resulto K=5 y J=1.

5.6.4.4. *Próxima Generación*

Parámetro	Descripción
Método de Selección	Se seleccionan los primeros individuos más aptos
Tipo de Cross over	Croosover multipunto a nivel de gen.
Probabilidad de Cross Over	65%
Cantidad Máxima de Genes	500
Cantidad Mínima de Genes	20
Probabilidad de reproducción	10%
Individuos Mantenidos	15%
Probabilidad gen nuevo	10%

5.6.4.5. *Conclusión*

Con esta configuración volvemos a tener el mismo problema que en la anterior, es principalmente, el tiempo de respuesta. Los resultados obtenidos son muy buenos para aplicaciones donde los tiempos de respuestas satisfactorios pueden ser largos.

5.6.5. Implementación V la Versión Final

Luego de los anteriores intentos; y de verificar el tiempo y espacio utilizado por los algoritmos al aprender, o diagnosticar un caso; vemos necesario una mejora del algoritmo. Para esto utilizaremos la memoria implementada en la configuración III; pero de una manera más eficiente.
Para lograr esto operamos de la siguiente manera: Está claro que cada vez que se reciba un caso producirá que los atributos involucrados en este caso, digamos *Marca* y *Precio*, por dar un ejemplo, mejoren su consideración para el individuo. Es decir, el algoritmo deberá aprender que los atributos *Marca* y *Precio*, son de mayor importancia, y si lo que se quiere es que el algoritmo corrija un concepto erróneo, se le pasara el caso equivocado (una compra no ofertada por ejemplo) y deberá bajar la importancia de los atributos en cuestión.
Esto debe ser así siempre. Por esto en cada lección, según cuál sea el tipo (si dice que un caso es bueno o malo), se modificará en la memoria el recuerdo de este atributo de manera explícita. No dependiente de los operadores Gmr y Ger. Como en la configuración III.

5.6.5.1. *Individuos*

Los individuos constan de un cromosoma por atributo. Los genes representan operaciones matemáticas, más las constantes numéricas 0, 1 y -1. En cada iteración estas operaciones matemáticas se combinan para formar un numero que representa la importancia relativa del atributo.

5.6.5.2. *Cromosoma*

Los cromosomas están formados por funciones matemáticas sencillas, más los números 1, -1,0. De esta manera con el correr de las iteraciones se forman números, además, como uno de los operadores es la multiplicación, podemos hacer que un atributo invierta su valor en una iteración (multiplicando por –1) de esta manera, si al

comprar un articulo, éste pasa a tener una menor importancia relativa, seguramente aquellos individuos que hallan mutados sus cromosomas, o hallan crecido invirtiendo la importancia, serán los seleccionados.

De manera formal podemos representarlos de la siguiente manera:
Sea F biyectiva tal que:

$$\forall(i,j)(0 \le i <\# A \wedge 0 \le j <\# Ai) \Rightarrow \exists(k)(k \le (\sum_{h=0}^{\#A} \# A_h) \wedge F(Ai_{(j)}) = G_k)$$

(O sea que hay un gen para cada valor de cada posible atributo más los siguientes genes.)

G_1 tal que representa a la función constante 1.

G_{-1} tal que representa a la función constante -1.

G_0 tal que representa a la función constante 0.

G_+ tal que representa a la función +.

G_- tal que representa a la función –

$G_/$ tal que representa a la función /

G_* tal que representa a la función *.

G_r Tal que representa un Recuerdo de un atributo
Sea i un valor entero.

G_r es tal que

$$\forall(k)(0 \le k \le \sum_{h=0}^{\#A} \# A_h \Rightarrow \exists i \mid (G_r G_k) = i)$$

Es decir devuelve un valor para cada elemento de cada atributo posible.
De los cromosomas podemos decir formalmente que:

$$\forall(k)(0 \le k \le \# A \Rightarrow (\exists C_k \wedge \exists G_n \wedge G_n \in C_k))$$

Un cromosoma por atributo, con al menos, un gen.

5.6.5.3. Fitness

La función de fitness toma en cuenta los atributos que tiene el caso, más el recuerdo, multiplicado por el valor aprendido en el cromosoma correspondiente a los atributos en cuestión. Es decir
Fitness = k tal que

$$k = \sum_{j=0}^{\#AtributosCaso} (\mathrm{Gr}(\mathrm{A_j}) * \mathrm{Eval}(\mathrm{C_j}))$$

Es decir, el fitness resulta de sumar la evaluación de cada cromosoma correspondiente un atributo y multiplicarlo por el recuerdo, esto es así para cada atributo perteneciente a un *Caso*, estos valores sumados forman el fitness.

$$\forall(k)(0 \leq \# A \Rightarrow (\exists C_k \wedge \exists G_n \wedge G_n \in C_k))$$

5.6.5.4. *Próxima Generación*

Parámetro	Descripción
Método de Selección	Se seleccionan los primeros individuos más aptos
Tipo de Cross over	Croosover multipunto a nivel de gen.
Probabilidad de Cross Over	65%
Cantidad Máxima de Genes	500
Cantidad Mínima de Genes	20
Probabilidad de reproducción	10%
Individuos Mantenidos	15%
Probabilidad gen nuevo	10%

5.6.5.5. *Conclusión*

Esta configuración resultó ser la mejor de todas en tiempo y performance.

5.6.6. Conclusión

Creemos que la aproximación elegida es muy buena, debido a que permite aplicar la técnica evolutiva pura, sin entrar en un esquema híbrido de técnicas de aprendizaje.
Esto se debe principalmente al concepto de memoria; siempre es posible obtener la misma performance, utilizando una mayor cantidad de individuos y más iteraciones por lección; o sea, que la memoria es una manera de optimizar los tiempos del algoritmo. Por otro lado con respecto a los resultados obtenidos, debemos decir que los tiempos y requerimientos de los algoritmos genéticos son bastantes caros. Aunque esto se verá más adelante, en él capítulo VII, cuándo se muestren los distintos resultados, y las comparaciones con las otras técnicas.

5.7. Comentarios Finales

En este capítulo dimos una breve introducción sobre algoritmos genéticos. Mostramos cuáles fueron las implementaciones realizadas para nuestro agente, explicando en detalle, la evolución de las distintas implementaciones hasta llegar a la mejor versión.
Algoritmos genéticos permiten, a la vez, que se mejoren y entrenen los individuos en el tiempo especializándolos sin perder generalidad, ya que el algoritmo seguirá buscando nuevas soluciones no exploradas, mediante mutaciones y cruce.

Capítulo VI

6. Perfiles Utilizando Redes Neuronales

Redes neuronales son una de las herramientas más poderosas que existen, en lo que a aprendizaje de máquina respecta. Siempre es la técnica a probar cuándo no se tiene nada mejor[RUS/1995], además, redes neuronales son una técnica robusta, ampliamente probada y usada, y es excelente en dominios altamente complejos en los que interaccionan gran cantidad de variables, además, redes neuronales brindan un buen desempeño tanto, cuándo el dominio trata con variables discretas, como continuas, o cuándo el dominio es mixto.
Por estas razones creemos que es bueno testear redes neuronales para aprender perfiles dinámicos de usuarios, analizar su desempeño, y compararlo con otras técnicas robustas de aprendizaje.

6.1. Redes Neuronales de Tipo Biológico

Se estima que el cerebro humano contiene más de cien mil millones (10^{11}) de neuronas y (10^{14}) sinapsis en el sistema nervioso humano. Estudios sobre la anatomía del cerebro humano concluyen que hay más de 1000 sinapsis a la entrada y a la salida de cada neurona. Es importante notar que aunque el tiempo de conmutación de la neurona (unos pocos milisegundos) es casi un millón de veces menor que en los actuales elementos de las computadoras, ellas tienen una conectividad miles de veces superior que las actuales supercomputadoras.
El objetivo principal de las redes neuronales de tipo biológico, es desarrollar un elemento sintético para verificar las hipótesis que conciernen a los sistemas biológicos.

Las neuronas y las conexiones entre ellas (sinápsis) constituyen la clave para el procesado de la información. Si observamos la *Figura 6.1,* la mayor parte de las neuronas poseen una estructura de árbol llamadas dendritas, que reciben las señales de entrada, que vienen de otras neuronas a través de las uniones llamadas sinápsis. Algunas neuronas se comunican sólo con las cercanas, mientras que otras se conectan con miles.

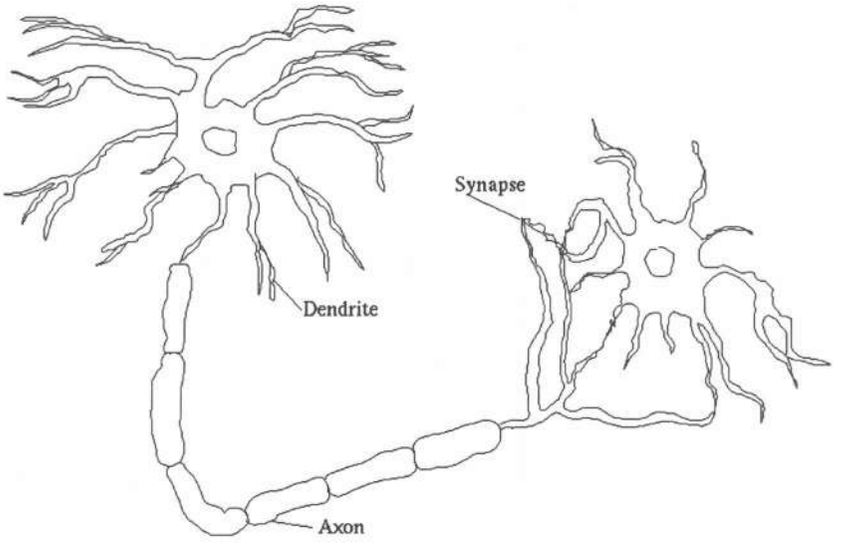

Figura 6.1 - Neuronas e interconexión

Hay tres partes principales en una neurona:

- el cuerpo de la neurona,
- ramas de extensión llamadas dendritas para recibir las entradas
- Un axón que lleva la salida de la neurona a las dendritas de otras neuronas.

La forma en que dos neuronas interactúan no está totalmente conocida aun, dependiendo además de cada neurona. En general, una neurona envía su salida a otras por su axón. El axón lleva la información por medio de diferencias de potencial, u ondas de corriente, que depende del potencial de la neurona. Este proceso es a menudo modelado como una regla de propagación representada por la función de red $u(.)$.La neurona recoge las señales por su sinápsis sumando todas las influencias excitadoras e inhibidoras. Si las influencias excitadoras positivas dominan, entonces la neurona da una señal positiva y manda este mensaje a otras neuronas por sus sinápsis de salida.

6.2. Redes Neuronales Artificiales

Como una analogía de las redes neuronales biológicas, las neuronas se modelan mediante unidades de proceso. Cada unidad de proceso se compone de una red de conexiones de entrada, una función de red (de propagación), encargada de computar la entrada total, combinada de todas las conexiones, un núcleo central de proceso,

encargado de aplicar la función de activación, y la salida, por dónde se transmite el valor de activación a otras unidades.

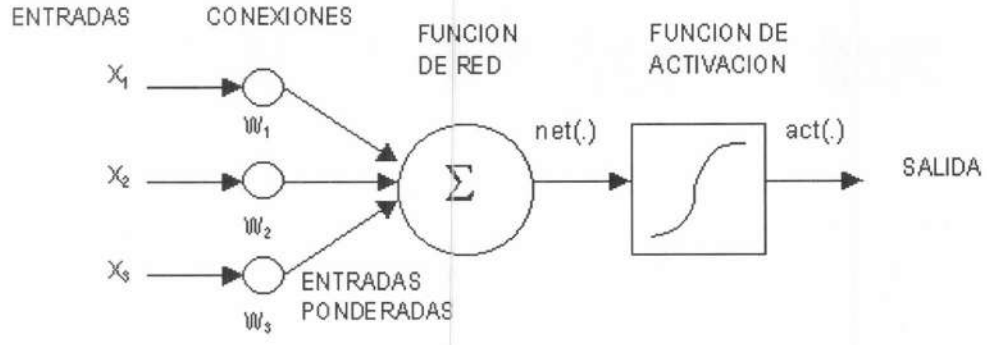

Figura 6.2 - Unidad de proceso típica

La función de red es típicamente la sumatoria ponderada, mientras que la función de activación suele ser alguna función de umbral o una función sigmoidal (La cual se detalla más adelante).

- *Función de propagación, o de red:* Calcula el valor de base o entrada total a la unidad, generalmente como simple suma ponderada de todas las entradas recibidas, es decir, de las entradas multiplicadas por el peso o valor de las conexiones. Equivale a la combinación de las señales excitatorias e inhibitorias de las neuronas biológicas.
- *Función de activación*: Es quizás la característica principal o definitoria de las neuronas, la que mejor define el comportamiento de la misma. Se usan diferentes tipos de funciones, desde simples funciones simples de umbral a funciones no lineales. Se encarga de calcular el nivel o estado de activación de la neurona en función de la entrada total. Es decir si la sumatoria de señales excitatoria, o alguna función aplicada a ella, sobrepasa algún cierto umbral.
- *Conexiones ponderadas:* Hacen el papel de las conexiones sinápticas, el peso de la conexión equivale a la fuerza o efectividad de la sinápsis. La existencia de conexiones determina si es posible que una unidad influya sobre otra, el valor de los pesos y el signo de los mismos definen el tipo (excitatorio / inhibitorio) y la intensidad de la influencia.
- *Salida:* calcula la salida de la neurona en función de la activación de la misma, aunque normalmente no se aplica más que la función identidad, y se toma como salida el valor de activación. El valor de salida cumpliría la función de la tasa de disparo en las neuronas biológicas.

Redes neuronales Biológicas	Redes neuronales Artificiales
Neuronas	Unidades de proceso
Conexiones sinápticas	Conexiones ponderadas
Efectividad de las sinápsis	Peso de las conexiones
Efecto excitatorio o inhibitorio de una conexión	Signo del peso de una conexión
Efecto combinado de las sinápsis	Función de propagación o de red
Activación -> tasa de disparo	Función de activación -> Salida

Tabla 6.1 - Comparación entre las neuronas biológicas reales y las unidades de proceso artificiales

6.2.1. Estructura y Formas de Interconexión

Para diseñar una red debemos establecer cómo estarán conectadas unas unidades con otras, y determinar adecuadamente los pesos de las conexiones. Lo más usual es disponer las unidades en forma de capas, pudiéndose hablar de redes de una, de dos, o de más capas, llamadas redes multicapa. Aunque inicialmente se desarrollaron redes de una sola capa, lo más usual es disponer tres o más capas: La primera capa actúa como buffer de entrada, almacenando la información bruta suministrada a la red o realizando un sencillo pre-proceso de la misma, la llamamos capa de entrada; otra capa actúa como interfaz o buffer de salida, almacenando la respuesta de la red para que pueda ser leída, la llamamos capa de salida; y las capas intermedias, principales encargadas de extraer, procesar y memorizar la información, las denominamos capas ocultas.

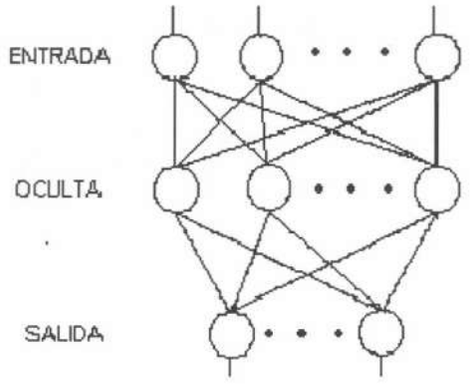

Figura 6.3 - Modelo de red en cascada de 3 capas

Además del número de capas de una red, en función de cómo se interconectan unas capas con otras, podemos hablar de *redes recurrentes* (feed-back), y redes no recurrentes o *redes en cascada* (feed-forward). En las redes en cascada la información fluye unidireccionalmente de una capa a otra (desde la capa de entrada a las capas

ocultas, y de éstas a la capa de salida), y además, no se admiten conexiones intracapa. En las redes recurrentes la información puede volver a lugares por los que ya había pasado, formando ciclos, y se admiten las conexiones intracapa (laterales), incluso de una unidad consigo misma.
Las conexiones entre una capa y otra pueden ser totales, es decir, que cada unidad se conecta con todas las unidades de la capa siguiente, o parciales, en las cuales una unidad se conecta con sólo algunas de las capas de la unidad siguiente, generalmente siguiendo algún patrón aleatorio o pseudo-aleatorio (por ejemplo, mediante algoritmos genéticos).
Desde una aproximación temporal, se puede distinguir además, entre conexiones sin retardo, y conexiones con retardo. Esto permite modelar aspectos dinámicos, por ejemplo para modelos psicofisiológicos de memoria.

6.2.2. La Función de Propagación (de red o de base)

Como ya hemos comentado, se encarga de calcular la entrada total de la neurona, como combinación de todas las entradas. La función más utilizada es la función lineal de base (LBF), que consiste en la sumatoria ponderada de todas las entradas.
Función lineal de base (LBF): Consiste en el sumatorio de las entradas ponderadas. Se trata de una función de tipo hiperplano, esto es, de primer orden.
Dada una unidad *j*, y *n* unidades conectadas a ésta, si llamamos *X* al vector de entradas (que coincide con las salidas de las unidades de la capa anterior) y *Wj* al vector de pesos de las conexiones correspondientes, esta función quedaría así

$$net_j(X, W_j) = \sum_{i=1}^{n} x_i w_{ij}$$

Al representar los pesos, utilizamos dos subíndices para indicar que conectan dos unidades, *i* y *j*, dónde *j* se refiere la unidad actual.
Función radial de base (RBF): Función de tipo hiperesférico, de segundo orden, no lineal. El valor de red representa la distancia a un determinado patrón de referencia.

$$net_j(X, W_j) = \sqrt{\sum_{i=1}^{n} (x_i - w_{ij})^2}$$

6.2.3. La Función de Activación

Se suele distinguir entre funciones lineales, en las que la salida es proporcional a la entrada; funciones de umbral, en las cuales la salida es un valor discreto (típicamente binario 0/1) que depende de sí la estimulación total supera o no un determinado valor de umbral; y funciones no lineales, no proporcionales a la entrada.
Casi todos los avances recientes en conexionismo, se atribuyen a arquitecturas multicapa, que utilizan funciones de activación no lineales, como una función de umbral, una gaussiana, ó en la mayoría de los casos, una función sigmoidal[QUI/1991]. El problema de trabajar con modelos no lineales radica en que son difíciles de describir en términos lógicos o matemáticos convencionales [RUM/1986].
Función de umbral: En un principio se pensó que las neuronas usaban una función de umbral, es decir, que permanecían inactivas y se activaban sólo si la estimulación total

superaba cierto valor límite; esto se puede modelar con una función escalón: la más típica es el escalón unitario: la función devuelve 0 por debajo del valor crítico (umbral) y 1 por encima.
Después se comprobó que las neuronas emitían impulsos de actividad eléctrica con una frecuencia variable, dependiendo de la intensidad de la estimulación recibida, y que tenían cierta actividad hasta en reposo, con estimulación nula. Estos descubrimientos llevaron al uso de funciones no lineales con esas características, como la función sigmoidal, con un perfil parecido al escalón de una función de umbral, pero continua.
Función sigmoidal o logística: Es probablemente la función de activación más empleada en la actualidad.

$$act_j(net_j) = \frac{1}{(1+e^{-net_j/\sigma})}$$

Se trata de una función continua no lineal con bastante plausibilidad fisiológica. La función sigmoidal posee un rango comprendido entre 0 y 1. Esto, aplicado a las unidades de proceso de una red neuronal artificial significa que, sea cuál sea la entrada, la salida estará comprendida entre 0 y1.
Esta función depende del parámetro s, que usualmente toma el valor 1. La salida de una unidad vale 0.5 cuando la entrada es nula, esto significa que la unidad tiene cierta actividad aun en ausencia de estimulación. Al aumentar la estimulación la unidad aumenta su activación, y la disminuye si la estimulación es inhibitoria, de forma parecida a cómo se comportan las neuronas reales.
Presenta las siguientes características deseables [WAS/1989].

1. Acomodación de señales muy intensas sin producir saturación.
2. Admite señales débiles sin excesiva atenuación
3. Fácilmente derivable, ya que:

$$f'(x) = f(x)\cdot(1-f(x))dx$$

La principal limitación de esta función es que no sirve para expresar polaridades, da siempre valores positivos. Una función alternativa con cualidades parecidas pero con un rango entre -1 y 1 es la función tangente hiperbólica. Desde un punto de vista fisiológico, el signo negativo se puede interpretar como una disminución de la tasa de disparo de una neurona por debajo de la tasa de disparo en reposo.
Función gausiana:

$$act_j(net_j) = c\cdot e^{-net_j^2/\sigma^2}$$

6.2.3.1. Redes Síncronas vs. Asíncronas

Un último aspecto no comentado, referido a la manera en que las unidades computan su activación con relación al tiempo, es que, si en cada ciclo de aprendizaje se calcula la activación de todas las unidades de una capa, hablamos de **redes síncronas**, en caso contrario se habla de **redes asíncronas** o **probabilísticas**, en las cuales cada unidad de proceso tiene una cierta posibilidad de computar su activación en cada ciclo de aprendizaje. Las redes asíncronas ofrecen la ventaja de ser menos propensas a caer

en mínimos locales (ver el problema de los mínimos locales en la Valoración de las redes de retropropagación).

6.2.3.2. Métodos Deterministas vs. Estadísticos

Esta distinción tiene bastante relación con la anterior, ya que las redes síncronas suelen emplear reglas de aprendizaje deterministas, y los métodos estadísticos en cambio, se aplican en las redes asíncronas. Algunos autores hablan de métodos de aprendizaje deterministas haciendo referencia al modo en que computan, paso a paso, las modificaciones en los pesos de las conexiones. Los métodos estadísticos hacen cambios pseudoaleatorios en las conexiones, y retienen los cambios sólo si mejoran la respuesta del sistema, pueden resultar útiles debido a que permiten alcanzar soluciones globalmente óptimas, a diferencia de las soluciones localmente óptimas típicas de los métodos deterministas.

6.3. Caracteristicas y Modo de Operación de las Redes Neuronales

6.3.1. Modo de Operación y Tipos de Aprendizaje

En cualquier red neuronal cabe distinguir la fase o proceso de aprendizaje, opcionalmente una fase de prueba, y la fase de aplicación (*recall*).
El aprendizaje consiste en la presentación de patrones a la red, y la subsiguiente modificación de los pesos de las conexiones siguiendo alguna regla de aprendizaje que trata de optimizar su respuesta, generalmente, mediante la minimización del error o la optimización de alguna "función de energía". El modo de aprendizaje más sencillo consiste en la presentación de patrones de entrada junto a los patrones de salida deseados (*targets*) para cada patrón de entrada, por eso se llama aprendizaje supervisado. Si no se le presentan a la red los patrones de salida deseados, diremos que se trata de aprendizaje no supervisado, ya que no se le indica a la red que resultados debe dar, sino que se le deja seguir alguna regla de auto-organización. Un tercer tipo de aprendizaje, a medio camino entre los anteriores, es el llamado aprendizaje por refuerzo, en este caso el supervisor se limita a indicar si la salida ofrecida por la red es correcta o incorrecta, pero no indica que respuesta debe dar.
Cualquiera que sea el tipo de aprendizaje usado, una característica esencial de la red es la regla de aprendizaje usada, que indica cómo se modifican los pesos de las conexiones en función de los datos usados en la entrada, es decir, de la historia de aprendizaje de la red. Por ejemplo, entre los algoritmos de aprendizaje supervisado, la *regla delta generalizada* modifica los pesos realizando en cada ciclo de aprendizaje un incremento de los pesos proporcional a la tasa de variación del error respecto al peso, en sentido negativo.
El aprendizaje requiere la presentación repetida de un número relativamente amplio de ejemplos de entrenamiento hasta lograr una respuesta satisfactoria del sistema (según la medida de error o función de energía utilizada).
Aunque la capacidad de aprendizaje es esencial y casi definitoria de lo que sería una red neuronal artificial, se han desarrollado modelos que no requieren modificar los pesos de las conexiones, sino que son precalculados y pre-establecidos antes de presentarle patrones a la red, como en las redes de Hamming y en redes de Hopfield.

6.3.1.1. Aprendizaje Supervisado

Ya sabemos que en el aprendizaje supervisado se presentan a la red una serie de patrones de entrada junto a los patrones de salida deseados. El aprendizaje consiste en la modificación de los pesos de las conexiones en el sentido de reducir la discrepancia entre la salida obtenida y la deseada. Los pasos generales a seguir por este algoritmo son los siguientes:

1. Aleatorizar los pesos de todas las conexiones (preferiblemente con valores pequeños)
2. Seleccionar un par de entrenamiento, es decir, un patrón de entrada, y el patrón de salida, deseado (target) correspondiente.
3. Presentar el patrón de entrada y calcular la salida de la red mediante las operaciones usuales: sumatoria de las entradas ponderadas, función de activación y transferencia a la siguiente capa, hasta llegar a la capa de salida. (inicialmente obtenemos salidas aleatorias, ya que los pesos de las conexiones son aleatorios)
4. Cálculo del error o discrepancia entre la salida obtenida y la deseada. El error (*función objetivo*) se suele definir cómo la suma de los cuadrados de las diferencias entre las salidas reales obtenidas y las deseadas, promediado para todas las unidades de salida y todos los patrones de entrenamiento. Si el error es menor de cierto criterio fijado de antemano, incrementar el número de ejemplos correctos; si todos los ejemplos se han clasificado correctamente, finalizar, sino, sino continuar.
5. Aplicar la regla de aprendizaje, es decir, ajustar los pesos de las conexiones tratando de disminuir el error, generalmente mediante el cálculo de tasas de variación o gradientes del error, por eso hablamos de reglas de aprendizaje por gradiente descendiente. Para reducir el error habrá que modificar los pesos de las conexiones, en proporción a la tasa relativa de variación del error con respecto a la variación del peso, o sea, la derivada del error respecto al peso, EP (error respecto al peso). Una forma de calcular el EP sería perturbar levemente un peso y observar cómo varía el error, pero no resultaría eficiente si trabajamos con muchas conexiones (este es el fundamento de algunos *métodos estadísticos* comentados en otro apartado). Más adelante veremos un algoritmo matemático para calcular EP.
6. Volver al paso 2

El aprendizaje supervisado se usa fundamentalmente en los *asociadores de patrones* o *memorias heterosasociativas* (ver más adelante). Al principio se aplicó el aprendizaje supervisado en redes con una sola capa de conexiones, como el *Perceptrón* [ROS/1962]

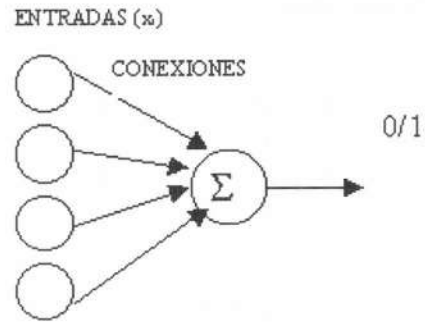

Figura 6.4 - Perceptrón de salida binaria

En los primeros perceptrones, con salida binaria, el aprendizaje era tan sencillo como sumar o restar las entradas a los pesos de las conexiones, dependiendo de, si la salida incorrecta era un 0, o un 1 respectivamente.
Pronto se elaboró una generalización para salidas continuas, conocida como regla delta: La modificación de cada peso se calcula sumándole la discrepancia entre la salida deseada y la obtenida, multiplicado por la entrada y por un coeficiente de aprendizaje:

$$w(t+1) = w(t) + \gamma \cdot x_i \cdot (Target - Out)$$

En los años 80 se popularizaron el algoritmo de retropropagación y la regla delta generalizada, que permiten entrenar redes no lineales de dos o más capas, mediante la propagación hacia atrás (hacia las capas ocultas) del error. La mayoría de las reglas empleadas actualmente en el aprendizaje supervisado son modificaciones de la regla delta generalizada. Se calcula que el 80% de las aplicaciones tecnológicas en redes neuronales se fundamentan en el algoritmo de retropropagación y en alguna versión de la regla delta generalizada.

6.3.1.2. Aprendizaje no Supervisado

Una de las críticas principales al aprendizaje supervisado, dejando de lado los aspectos computacionales, era su dudosa plausibilidad biológica, pues:

a. Requiere propagar cierta información (el error cometido) hacia atrás, en sentido contrario al flujo natural de la información. Como es obvio, esto no sucede en las neuronas reales, sin embargo esta objeción es superficial si tenemos en cuenta que el cerebro es rico en redes que retrogradan de unas capas a otras, y podría utilizar estos caminos de múltiples formas para aportar la información del tipo que el aprendizaje supervisado requiere.
b. Requiere de un instructor que proporcione la salida deseada. Las personas aprendemos a comprender frases o escenas visuales sin instrucciones explícitas para ello, se podría decir que el cerebro posee algún mecanismo de auto-organización. Cuando se le presentan a la red una gran colección de datos sin indicarle que debe hacer con ellos, se encuentra aparentemente sin saber que

hacer. No obstante, los investigadores han diseñado varios procedimientos de aprendizaje no supervisado, que son capaces de extraer inductivamente suficiente información de esos datos como para permitir posteriormente su recuperación, estos algoritmos son utilizados en tareas de clustering y clasificación.

En el aprendizaje no supervisado no se requiere presentar patrones de salida deseados. Sin embargo, el algoritmo y la regla de modificación de las conexiones producen vectores de salida consistentes, esto es, la presentación de un patrón aprendido o parecido a él, produce siempre la misma salida. Se considera que el proceso de aprendizaje es capaz de extraer ciertas propiedades estadísticas de los ejemplos de aprendizaje y agruparlos en categorías o clases de patrones similares. No sabemos a priori que salida corresponderá a cada tipo o grupo de patrones de entrada, ni que atributos usará para clasificarlos, por eso son muy útiles como generadores de categorías (clustering)
En general, los métodos de aprendizaje no supervisado apelan a alguna noción de la calidad de la representación. Las representaciones de calidad se caracterizan por la economía de la representación (el costo de codificación más el costo de reconstrucción), sin perder la capacidad de reconstruir la información almacenada.

6.3.1.3. Aprendizaje por Componentes Principales

El aprendizaje por componentes principales consiste en extraer, y describir, los componentes de variación compartidos por multitud de unidades de entrada, siendo suficiente una unidad para cada componente principal. Se puede lograr haciendo que las redes reconstruyan la entrada sobre unas unidades de salida, aplicando retropropagación. Este aprendizaje no es supervisado, porque dado que la salida es la misma que la entrada, no se necesita instructor. Otra forma de lograrlo sin usar retropropagación se basa en modificar las conexiones en función de las correlaciones entre las actividades de las unidades ocultas y las unidades de entrada.

6.3.1.4. Aprendizaje Competitivo

En el aprendizaje por componentes principales, un pequeño número de unidades coopera en la representación del patrón de entrada. En el aprendizaje competitivo, un gran número de unidades ocultas, pugnan entre sí, con lo que finalmente se utiliza una sola unidad oculta para representar un patrón de entrada determinado, o un conjunto de patrones similares. La unidad oculta seleccionada es aquella cuyos pesos incidentes se asemejan más al patrón de entrada. El aprendizaje consiste en reforzar las conexiones de la unidad ganadora, y debilitar las otras, para que los pesos de la unidad ganadora se asemejen cada vez más al patrón de entrada. La reconstrucción del patrón de entrada consistirá en copiar la distribución de pesos de la unidad ganadora.

6.3.1.5. Aprendizaje por Refuerzo

Como ya comentamos, la idea es similar a la del aprendizaje supervisado, sólo que aquí la información dada por el maestro es mínima, se limita a indicar si la respuesta de la red es correcta o incorrecta. En este sentido, se asimila a la noción tomada de la

sicología conductual de condicionamiento por refuerzo, que en resumen defiende que se aprenden (en el sentido de que tienen más probabilidad de repetirse) las conductas reforzadas positivamente y viceversa (las conductas castigadas o reforzadas negativamente reducen la posibilidad de aparecer).
Una noción estrechamente ligada a la de aprendizaje reforzado es la de redes basadas en la decisión. Son redes para clasificación de patrones, similares a los asociadores de patrones, en las cuales el maestro, en vez de indicar la salida deseada exacta (target), indica sólo la clase correcta para cada patrón de entrada.
En las redes basadas en decisión se utiliza una función discriminante para determinar a que categoría o clase pertenece el patrón de entrada, si durante el entrenamiento, una respuesta no coincide con la dada por el maestro, se cambian los pesos, si no, se dejan igual. El aprendizaje consiste en encontrar los pesos que dan la clasificación correcta.

6.3.2. Aprendizaje - Redes de Retropropagación

Al hablar de redes de retropropagación, o redes de propagación hacia atrás hacemos referencia a un algoritmo de aprendizaje, más que a una arquitectura determinada. La retropropagación consiste en propagar el error hacia atrás, es decir, de la capa de salida hacia la capa de entrada, pasando por las capas ocultas intermedias y ajustando los pesos de las conexiones con el fin de reducir dicho error. Hay distintas versiones o reglas del algoritmo de retropropagación, y distintas arquitecturas conexionistas a las que pueden ser aplicados.
Durante mucho tiempo no se dispuso de algoritmos para entrenar redes multicapa, y como las redes de una capa estaban muy limitadas en cuanto a lo que eran capaces de representar, el campo de las redes neuronales artificiales estaba estancado. La invención y perfeccionamiento del algoritmo de retropropagación dio un gran impulso al desarrollo de este campo. Tiene un buen fundamento matemático, y a pesar de sus limitaciones, ha expandido enormemente el rango de problemas donde se aplican las redes neuronales artificiales.

6.3.2.1. Descripción Matemática del Algoritmo de Retropropagación

Se explica una versión del algoritmo [HIN/1992] para redes con las siguientes características:

- No recurrentes
- Función de activación sigmoidal
- Capas totalmente interconectadas
- Operación totalmente síncrona

Algoritmo:

1. Aleatorizamos los pesos de las conexiones.
2. Presentamos un patrón de entrada y calculamos la salida.
3. Dada una unidad j-ésima de la capa de salida, y unidades i-ésimas de la capa oculta inmediatamente anterior, calculamos la entrada total ponderada, y la salida o activación de la misma.

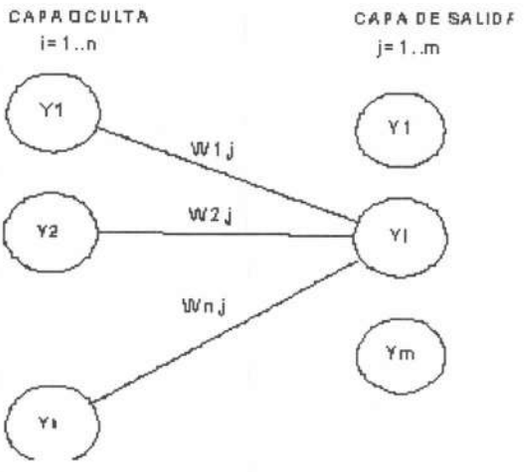

Figura 6.5 - Calculo de entrada ponderada del nodo Yj

4. Una vez computadas las actividades de todas las unidades de salida, se calcula una estimación del error, generalmente una función cuadrática de los errores individuales cometidos por cada unidad, siendo cada error individual la diferencia entre la salida deseada y la obtenida.

$$E = \frac{1}{2}\sum_{j}(d_j - y_j)^2$$

Siendo dj la salida deseada para la unidad j-ésima.
Se van a indicar por un lado las expresiones matemáticas y por otro lado la explicación intuitiva de cada paso. Conviene recordar que nuestro objetivo es calcular cómo varía el error al variar el peso de cada conexión (tasa de variación del error respecto al peso de una conexión, EP)

$$x_j = \sum_{i=1}^{n}(y_i \cdot w_{ij}) \qquad y_j = \frac{1}{1+e^{-x_j}}$$

5. Cómputo de la rapidez de variación del error al cambiar la actividad de cada unidad de salida (EA, error respecto a la actividad)

$$EA_j = \frac{\partial E}{\partial y_j} = d_j - y_j$$

Es justamente la diferencia entre la salida deseada y la salida real obtenida, es decir, la diferencia entre la actividad deseada y la actividad real

6. Cómputo de la rapidez de variación del error al cambiar la entrada total que recibe cada unidad de salida.

$$\delta_j = \frac{\partial E}{\partial x_j} = \frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial x_j} = EA_j y_j(1-y_j) = y_j(1-y_j)(d_j - y_j)$$

Es igual a la tasa de variación del error, al variar su activación multiplicado por la tasa de variación de la activación al cambiar su entrada (que es justamente la derivada de la función sigmoidal)

7. Cómputo de la rapidez de variación del error al ser modificado un peso de la conexión aferente a una unidad de salida.

$$EP_{ij} = \frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial x_j}\frac{\partial x_j}{\partial w_{ij}} = \delta_j y_i$$

Es igual a la tasa de variación del error al variar su entrada, por la tasa de variación de la entrada al variar ese peso.
Hasta ahora sabemos calcular el EA, sólo para las unidades de salida, ¿ qué pasa con las unidades ocultas?. En este caso no tenemos una estimación directa del error aportado por cada unidad oculta; aquí es donde interviene la retropropagación o propagación hacia atrás del error:

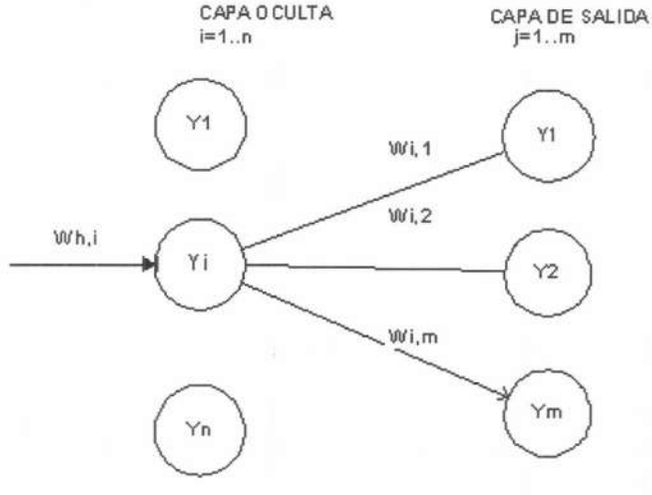

Figura 6.6 – Efecto de una capa oculta sobre las unidades de salida

La unidad i-ésima de la capa oculta afecta a todas las unidades de salida, por lo tanto, para estimar cómo varía el error al variar la actividad de esa unidad oculta, habrá que sumar los efectos individuales de su actividad sobre todas las neuronas de salida. Cada efecto individual sobre la variación del error, será igual a la tasa de variación del error de la unidad de salida al cambiar su entrada total, multiplicado por la tasa de variación de su entrada al variar la actividad de la unidad oculta.

$$EA_i = \frac{\partial E}{\partial y_i} = \sum_{j=1}^{m} \left(\frac{\partial E}{\partial x_j} \frac{\partial x_j}{\partial y_i} \right) = \sum_{j=1}^{m} \left(\delta_j \cdot w_{ij} \right)$$

8. Conociendo EA para las unidades de cualquier capa podemos calcular d y EP con las expresiones ya conocidas.

$$\delta_i = EA_i y_i (1 - y_i) = y_i (1 - y_i) \sum_{j=1}^{m} \left(\delta_j w_{ij} \right)$$

$$EP_{ik} = \delta_i \cdot y_k$$

9. Disponiendo de la tasa de variación del error respecto al peso de una conexión (EP), podemos usar distintas reglas para modificar ese peso con el fin de reducir dicho de error. Una de las primeras reglas que aprovechó este algoritmo es la regla delta generalizada, que calcula el incremento a aplicar a un peso como una proporción directa de la tasa de variación del error.

$$\Delta w_{ij} = \eta \delta_j y_i$$

siendo h el coeficiente de aprendizaje, típicamente con valores comprendidos entre 0.01 y 1.0

$$w_{ij} (t + 1) = w_{ij} (t) + \Delta w_{ij}$$

6.3.3. Propiedades de las Redes Neuronales

Destacamos a continuación una serie de atributos importantes de las redes neuronales, en los que a aprendizaje refiere. Estos no da una idea previa sobre cuál es el resultado que podemos esperar de ellas.

6.3.3.1. Sesgo

De representación: El sesgo de las redes neuronales en general viene dado por problemas del tipo Back propagation N dimensión basada en los pesos de espacios de hipótesis continuos, es decir, el sesgo surge de la misma regla de aprendizaje, de la arquitectura de la red, y de la forma en que se interconectan.

De búsqueda: Viene dado por la el trade off entre gradiente descendente y la forma de recorrer el espacio de búsqueda sobre la asignación de los pesos a la Red.

6.3.3.2. Problemas Apropiados

- Al ser las redes neuronales representaciones basadas en atributos, estas carecen de la capacidad de representar funciones lógicas generales. Aunque

teniendo varios niveles podemos representar una función de este tipo, para ello se puede llegar a necesitar 2^N bits para su especificación.

- Las redes neuronales son buenas para representar funciones continuas tanto en su entra da como en su salida.
- Reconocimiento de caracteres impresos mecánicamente.
- Reconocimiento de gráficos.
- Reconocimiento de caracteres escritos a mano.
- Reconocimiento de escritura manual cursiva.
- Análisis de olor y aroma.
- Perfilamiento de clientes en función de la compra.
- Desarrollo de productos.
- Control de Calidad.
- Distribución recursos hidráulicos para la producción eléctrica
- Ayuda al diagnóstico.
- Análisis de Imágenes.
- Desarrollo de medicamentos.
- Distribución de recursos.
- Optimización de rutas.
- Optimización en la distribución de recursos

6.3.3.3. Ruido

Las redes neuronales artificiales son capaces de extraer o abstraer las características esenciales de las entradas aprendidas, de esta manera pueden procesar correctamente datos incompletos o distorsionados.

6.3.3.4. Sobreaprendizaje

Puede surgir sobreaprendizaje, por ejemplo, en redes del tipo BackPropagation ya que estas podrían quedar en mínimos locales.

6.3.3.5. Otros

- Caja Negra: Podemos ver a una red neuronal como una caja negra. Ya que aunque el aprendizaje de la función halla sido optimo, no sabremos porque.
- Conocimiento Previo: Debido a la falta de transparencia es que se dificulta la utilización de conocimiento previo para la preparación de la red.
- Generalización: Suele ser buena para los casos en que la correlación entre atributos es baja y la función a aprender varia de manera gradual.
- Entrenamiento**:** Tiene un entrenamiento bastante lento, aunque en general presenta una respuesta rápida ante la consulta en una red ya entrenada.
- *Memoria distribuida*: el conocimiento acumulado por la red se halla distribuido en numerosas conexiones, esto tiene como consecuencia la *tolerancia a fallos:* una red neuronal es capaz de seguir funcionando adecuadamente a pesar de sufrir lesiones con destrucción de neuronas o sus conexiones, ya que la información se halla distribuida por toda la red, sin embargo en un programa tradicional un pequeño fallo en cualquier punto puede invalidarlo todo, y dar un resultado absurdo o no dar ningún resultado.

6.4. Redes Neuronales Para Generar Perfiles Dinámicos de Usuarios

Los tipos de redes neuronales existentes son muchos, y por sobre todas las cosas, muy variados. La cantidad de opciones a implementar es muy amplia. Optamos aquí por implementar una de las más conocidas y probadas. El Perceptron multicapa con back-propagation como técnica de adaptación. Éste ha probado su eficiencia en problemas de reconocimientos de patrones y clustering.
Mostramos cómo se implementó la técnica de redes neuronales en nuestro agente; y cómo lo utilizamos para aprender y mantener un perfil del usuario. La idea de la implementación final es la de trasladar cada uno de los valores posibles de los atributos discretos, a un nodo de entrada de la red neuronal, de igual manera, se trasladan los atributos continuos a un nodo que lo representa. Luego, se tienen dos capas ocultas, y una capa final de un nodo, cuya salida, es un valor real que representa el interés despertado al usuario por el producto.
Como dijimos anteriormente la técnica utilizada es supervisada, y el agente hace las veces de maestro de la red. Si alguno de estos conceptos no le es familiar es conveniente remitirse a la sección anterior, o a la bibliografía en especial a estos textos [HER/1990], [MIT/1997], [RUS/1995], [QUI/1991], [RUM/1986], [WAS/1989] citados en orden de importancia.
Para la implementación del aprendizaje con redes neuronales, utilizamos un framework de distribución gratuita que brinda una interfaz para el desarrollo y aprendizaje de redes neuronales[JOO/2002]. Este framework permite crear varios tipos de redes neuronales. Las distintas configuraciones probadas giran siempre alrededor de la estructura de Perceptron multicapa con Back propagation.
A continuación mostraremos con un pequeño ejemplo como podemos configurar y utilizar una red neuronal para aprender un perfil de usuario; luego, detallaremos nuestra implementación, y cuál fue la codificación de las variables tenidas en cuenta para la implementación de esta técnica.

6.4.1. Representación de Perfiles por Medio de una Red Neuronal

En esta sección detallaremos cómo se puede utilizar una red neuronal para representar un perfil de usuario, cómo se hace para adaptar el perfil en el tiempo, y cómo utilizar el perfil, para diagnosticar si un determinado producto le gustará o no al usuario. Plantearemos un ejemplo sencillo, pero conceptualmente similar a la implementación realizada. La diferencia radica en la cantidad de atributos del dominio, en nuestro ejemplo, es mucho menor. Tomaremos un subconjunto del dominio, el cual se especifica en la *tabla 6.2*

Atributo	Tipo	Valores Posibles
Target	Discreto	niño dama hombre adolescente
Precio	Continuo	[0-10000]

Tabla 6.2 – Subconjunto de variables usado en el ejemplo de aprender un perfil, utilizando redes neuronales.

6.4.1.1. Ejemplo de Representación

Con el dominio especificado en la tabla 6.2 en mente, pasemos al problema de aprender un perfil, y cómo adaptarlo en el tiempo a los gustos del usuario, utilizando una red neuronal. Sabemos que un perfil está formado por atributos, y que la red se encargará de diagnosticar el grado de interés de estos atributos. Para esto debe poder recibir como entrada atributos e inferir la importancia de ellos para el usuario.

Representación de los atributos del dominio

Los atributos del dominio hay que codificarlos de manera que sean entendibles por la red neuronal. Esta codificación se realizará de la siguiente manera, según el tipo del atributo a codificar:
Para las variables discretas, habrá un nodo de entrada por cada valor posible del atributo, si el valor de atributo que representa el nodo no se conoce, el valor de entrada será cero, caso contrario el valor será 1. La función de activación es la función escalón con un umbral de 0.5, esto sirve para captar la presencia o no del atributo, emitiendo como salida del nodo un uno o un cero, según represente, o no el nodo al valor del atributo, respectivamente.
Para los atributos continuos, habrá un nodo por cada atributo, la función de activación es la sigmoidea, ya que no queremos perder las características continuas del atributo. Al ser la sigmoidea una fusión monótona y estrictamente decreciente en un entorno positivo, como es el caso de todos nuestros atributos continuos; nos asegura, que los distintos valores del atributo serán considerados, también, de manera distinta. Un ejemplo de esto es el atributo *CuandoCompro*, entre más tiempo pasó seguramente será mayor la probabilidad de compra. Buscamos captar esta propiedad con la representación elegida.
Ahora bien, ¿qué quiere decir esto, y cómo se representa?, Para esto analicemos el gráfico de la *Figura 6.7.*

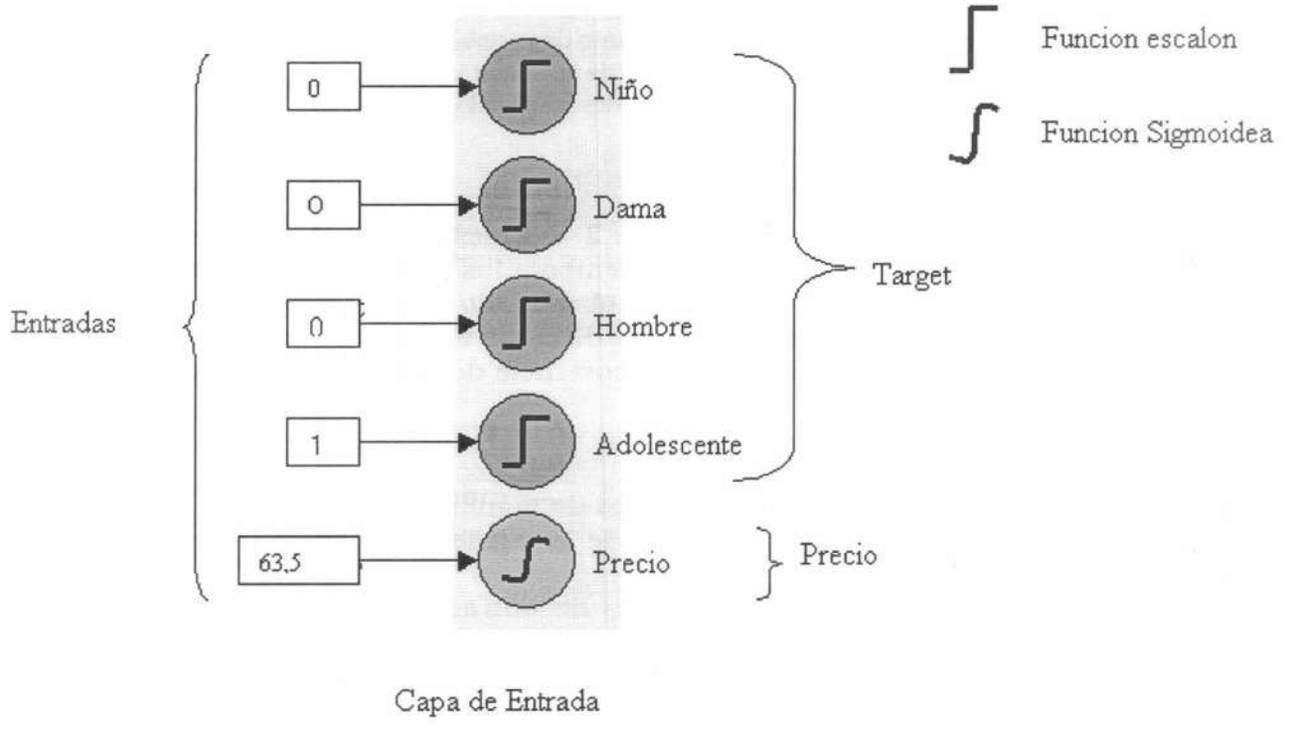

Figura 6.7 - Ejemplo de codificación correspondiente a la primer capa de la red neuronal

En la *Figura* 6.7 puede verse representado el dominio de nuestro agente, es decir el atributo *target*, discreto; y el atributo precio, continuo. Supongamos ahora, que queremos que la red diagnostique el valor relativo para el usuario, de un conjunto de atributos formados por los siguientes valores:

Atributos = {Target=Adolescente, Precio=63,4}

La capa de entrada emitirá un valor cero, para los primeros tres nodos; un valor uno, para el siguiente, que representa al valor *Adolescente* del atributo *target*; y para el último nodo, el valor 63.4 correspondiente, al atributo continuo precio. Estos datos figuran etiquetados con la palabra *entradas* en el gráfico anterior. La salida de estos nodos será, simplemente, la aplicación de la función correspondiente al nodo, o cero si no sobrepasan el umbral de activación. El umbral es de 0.5 para todos los nodos.
Sabemos que cualquier implementación de red neuronal deberá tener, capas de entrada, capas intermedias, y una capa final de salida. Nos falta decidir cómo serán, con qué criterio se agregan las capas intermedias; y cómo será la capa de salida.

Capas Intermedias u ocultas

Definiremos las capas intermedias de la siguiente manera. En la primer capa oculta habrá un nodo por cada atributo de la capa de entrada, de esta manera buscamos que cada nodo capte la relación con los valores de atributos que le corresponden de la capa de entrada si el atributo es discreto; a la vez, trata de captar correlaciones existentes entre los nodos de entradas. Si el atributo es continuo, se usará el nodo para obtener

relaciones con los demás atributos. Las demás capas ocultas, luego de la primera tendrán, la mitad de nodos que la capa inmediata anterior. Esta decisión es arbitraria, y busca captar la mayor cantidad de relaciones posibles entre los datos.
En todos los casos la función de activación será la sigmoidea. Este modelo se ve reflejado en la *Figura 6.8*

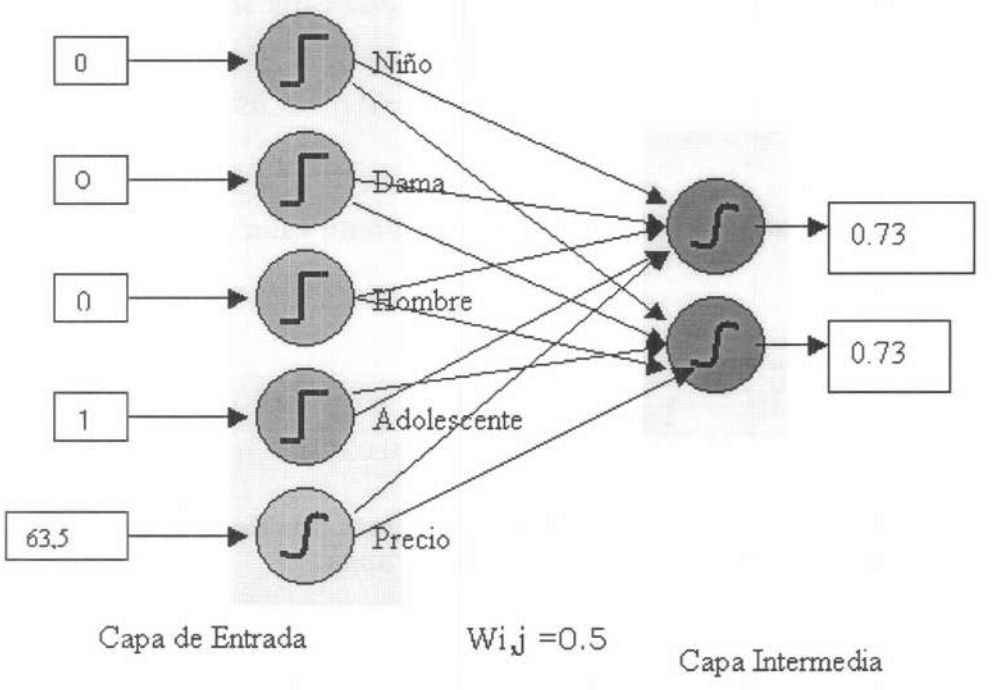

Figura 6.8 - Primeras dos capas de la red neuronal

En redes neuronales, normalmente, la cantidad de capas ocultas, y la cantidad de nodos en cada capa, es arbitraria; no existe manera exacta de calcular u obtener la configuración ideal, se pueden utilizar heurísticas e idea generales, como las que venimos aplicando, pero no hay métodos concretos e independientes del dominio, que solucionen este problema.
Para entender el modo en que se realiza la propagación hacia delante, que es el método ejecutado por la red para dar su diagnóstico sobre los atributos de entrada; podemos hacer los cálculos, y ver cuáles serían los valores de salida de la capa intermedia. Para simplificar los cálculos supondremos que el valor de los pesos de las conexiones es de 0.5 para todas, esto es Wij=0.5 para todo Wij, según la notación que venimos usando desde las secciones anteriores.
Para las primeras tres neuronas la salida es cero, ya que utilizan la función escalón con umbral en 0.5, para la neurona 4 el valor es uno, por la misma razón; y para la neurona 5 la salida es 0.99, que corresponde a aplicar la función sigma a la entrada

$$0.99 = \frac{1}{1+e^{-63.5}}$$

Luego por estar las neuronas conectas entre sí, y por ser los valores de las conexiones todos iguales, la entrada de las neuronas de la segunda capa será el mismo.
Este es igual a

$$Entrada(N2i) = 1 * 0.5 + 0.99 * 0.5 = 0.995$$

Aplicando la función sigma una vez más, nos el siguiente valor

Salida(N2,i) =

$$0.73 = \frac{1}{1+e^{-0.995}}$$

Que es el valor que aparece en la salida de las dos neuronas de la capa numero dos, en el gráfico; y que a su vez, será entrada de las otras capas.

Capa de salida

Por ultimo, la capa de salida, con una sola neurona, y que es la encargada de decir cuál es la importancia de un determinado conjunto de atributos, esta última también utiliza la función sigmoidea, debido a que necesitamos una función continua, que nos diga que tan buenos considera estos atributos el usuario. Justamente la función sigmoidea va de 0 a 1, lo que nos sirve para ranquear distintos conjuntos de atributos.

La red quedaría ahora de la siguiente manera.

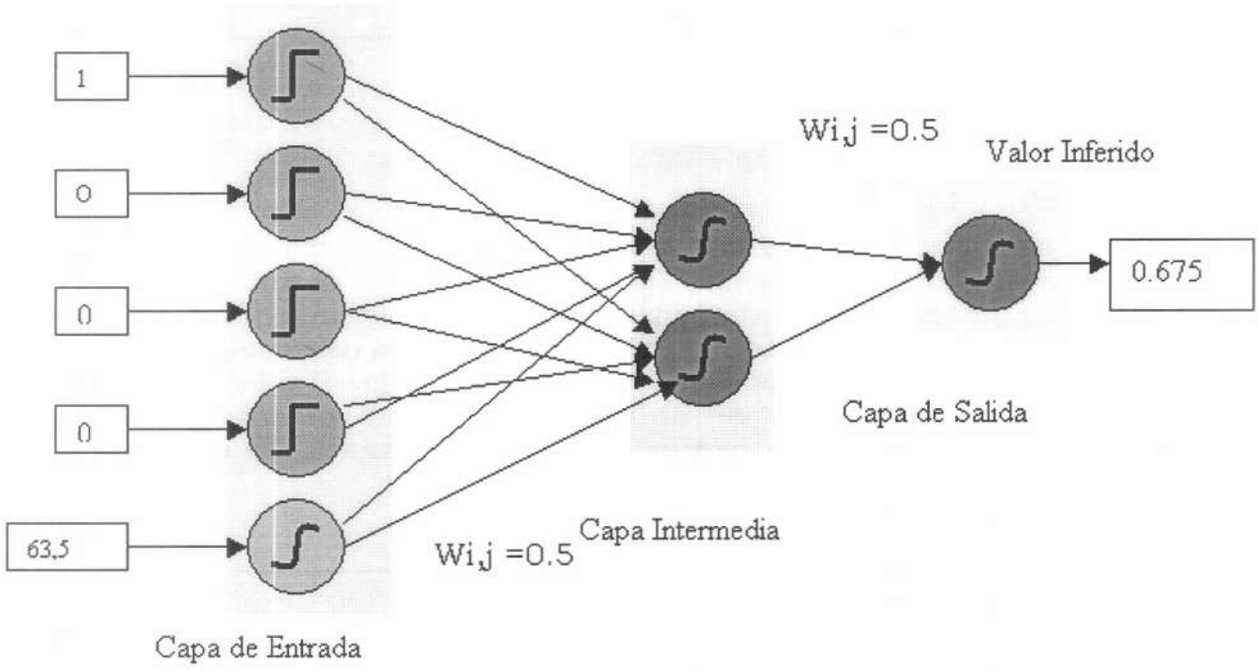

Figura 6.9 – Red neuronal utilizada para aprender un perfil de usuario con dos atributos

Este último valor se obtiene de aplicar la función sigma, a la entrada que es igual a 0,73; y el valor inferido será igual 0,675, que corresponde a la importancia de los datos de entrada para la red neuronal.
Con este ejemplo podemos notar, que en el caso de la red neuronal, la red toda es el perfil, esto incluye los pesos de las conexiones, y los umbrales de los nodos. Por ejemplo, si la conexión del nodo 1, con los nodos de la capa intermedia, tuviera un peso de 0.3 el resultado final (Valor inferido) sería, más alto que el inferido en el ejemplo. Es precisamente, en la adaptación y modificación de los pesos que la red, se adapta y se modifica, modificando el perfil.

6.4.2. Aprendizaje y Diagnóstico

6.4.2.1. Diagnóstico

Dar un diagnóstico sobre la importancia relativa de un conjunto de atributos, para la implementación realizada utilizando redes neuronales, es bastante sencillo; sólo se trata de codificar los atributos utilizando la heurística explicada en la sección anterior, y dárselas como entrada a la red neuronal. Esta realizará el cálculo de las entradas, y propagará el cálculo a través de todas las capas, y devolverá un valor real que se presenta la importancia de los atributos para el usuario (para más detalles remitirse a la sección 6.2.2 a lo largo del ejemplo anterior se muestra, paso a paso, cómo se realizan los cálculos en cada nodo y cómo se propagan los resultados de cada nodo hacia delante, es decir hacia la capa de salida.

6.4.2.2. Aprendizaje

Por el lado del aprendizaje, es algo más dificultoso. Para hacer que la red aprenda, tomamos la siguiente decisión. El agente, *Vendor,* hace las veces de tutor o maestro de la red neuronal. Entonces, si un determinado conjunto de atributos es considerado bueno, el agente pide una inferencia a la red neuronal. La red neuronal devolverá un valor, digamos *k*. Luego el agente, le dice a la red neuronal, que el valor *k* es de poca importancia para ese conjunto de atributos, y la red neuronal utilizando backpropagation (ver sección 6.3.3), corrige de manera automática, los valores de los pesos de los nodo.
Algorítmicamente realiza los siguientes pasos.

```
Metodo Aprender(atributos, valor, red neuronal)
Input
        Atributos: Atributos a los que se debe modificar la importancia.
        Valor:  Valor que debería haber diagnosticado la red neuronal
        Red Neuronal:       Red Neuronal
begin
        double res = RedNeuronal.diagnosticar(Atributos)
        if not (res < valor)
                RedNeuronal.backpropagation (atributos, valor)
End
```

Figura 6.10 – Algoritmo utilizado para el aprendizaje de nuestra implementación utilizando redes neuronales

La retropropagación tratará de modificar los pesos, de manera que la próxima vez que se le presente el mismo caso, la inferencia dada por la red sea la que el maestro le dictó.
El agente decide que valores se le pasará a la red neuronal para cada caso, en nuestra implementación este valor (parámetro del algoritmo anterior) es de 1 cuando la lección es tomada como positiva, y cero cuando la lección deba ser tomada como negativa.

6.5. Implementación

Como bien dijimos en la sección anterior, el agente funciona como tutor y maestro de la red neuronal, modificando la red, y por ende, modificando el perfil de usuario que esta red representa. El agente, por cada lección positiva que quiera dar a la red neuronal, realiza los siguientes pasos:

Sea y = ValorInferido (caso), donde caso, es el caso dado a la red para su inferencia
El agente ejecutara el método mostrado en la sección anterior con los siguientes parámetros: **Aprender** (*Caso,1, RN),* es decir le dice que el valor del caso debe ser 1, la red intentara aprender que el valor del caso es uno, utilizando backpropagation (ver

sección 6.3.2 para más detalles) y adaptado los pesos de las conexiones para que se den un valor lo más cercano posible a 1. Claro está que backpropagation no hará que la red retorne el valor 1 la próxima vez que se le presente el mismo caso, esto no pasará incluso después de muchas iteraciones, debido a la lentitud intrínseca de las redes neuronales para aprender.
De igual manera, si se le quiere dar la lección contraria es decir, que el caso es malo, se arma una lección con los siguientes datos *Caso =0,* es decir el valor del caso es 0, y Tratará de adaptar los pesos a este valor.
Otra aproximación de esto sería que, según la importancia de la lección se dé los distintos valores comprendidos entre 0 y 1, sin embargo eso fue probado y no reportó mayores modificaciones a la implementación más sencilla del algoritmo.

6.5.1. Codificación de las Variables

Unas de las tareas principales, como lo a sido en todas las implementaciones hechas hasta aquí, es elegir la codificación de las variables de manera que sean entendibles por la red neuronal. A continuación se detallan las decisiones tomadas, y que criterios se utilizaron para su elección. Los cuales son, en general, similares a los tomados en el ejemplo mostrado en la sección anterior.
Para la implementación del agente utilizando redes neuronales decidimos tratar a la variable continua fecha, como discreta, imitando la codificación que dimos para las redes bayesianas en el Capítulo IV, y para algoritmos genéticos en el capítulo V. Esto se debe, fundamentalmente, a que el tratamiento continuo de la fecha no reporta mayores beneficios. La compra en determinadas fechas es una función inarmónica, cuyos mínimos y máximos están bien definidos, y por sobre todas las cosas, es demasiado oscilante. Esto se ve reflejado en el tratamiento estadístico de los datos, el cual se resume en el apéndice A.
Los demás atributos continuos, como ser el atributo *CuandoCompro*, podríamos decir que tienen una incidencia casi lineal, por ejemplo, a medida que *CuandoCompro* es mayor, la probabilidad de que compre un producto es mayor. Para el atributo *Precio*, en cambio, depende un poco, hay individuos que prefieren comprar caro en relación con los precios de igual categoría; y eso podría justificar tratarlo como discreto, Igualmente, nosotros decidimos tratarlo como continuo, debido a que simplifica la cantidad de nodos de entrada para la red.

Nombre	Descripción	Tratamiento
Categorías	Categoría del producto	Discreto
Accesorios	Categoría accesorios	Discreto
Calzado	Categoría calzados	Discreto
Deportes	Categoría deportes	Discreto
Entrenamiento	Categoría entrenamiento	Discreto
Indumentaria	Categoría indumentaria	Discreto
Marcas	Categoría marcas	Discreto
CategoríaComprador	a quién está destinado el producto	Discreto
Precio	Precio del Producto	Continuo
CatPrecio	categoría del precio	Discreto
Exposición	tiempo de exposición	Continuo
Dia	Día de la semana	Discreto
Mes	Mes del año	Discreto
Semana	Semana del Mes	Discreto
Color	Colores	Discreto
RankingInterno	Categorización del desempeño del producto dentro de la empresa	Discreto
CuandoCompro	tiempo transcurrido desde la compra	Continuo
Calidad	Calidad del producto	Discreto
Talle	Talle del producto	Discreto
Interés	Concepto objetivo interés dado el producto	Continuo

Tabla 6.3 - Codificación de las variables del perfil

6.5.2. Arquitectura de la Red

Durante la construcción de esta red neuronal, no hubo un avance paulatino, sino varios intentos, y varias redes creadas, antes de dar con una implementación cuya performance sea aceptable para el dominio en que se desempeñará. El problema principal en casi todos los casos, se debió al tiempo de entrenamiento excesivamente largo, e intolerable para este tipo de aplicaciones.
Todas las implementaciones tenidas en cuenta giraron en torno a la misma codificación de las variables descriptas en el ejemplo anterior. Lo que fuimos variando fue la cantidad de capas intermedias, y la cantidad de nodos intermedios. La metodología aplicada para encontrar la red que se explica más adelante fue: Partimos de una red con 6 capas, y la siguiente estructura. (96,48,24,12,6,1), donde los 96 nodos de la primer capa corresponde a la codificación de atributos explicada en la sección anterior. Los resultados empíricos de estas no eran malos, a no ser, porque tardaba más de 10 minutos para realizar algunas operaciones. Luego fuimos disminuyendo la cantidad de nodos intermedios y de capas, hasta llegar a un punto en el que las soluciones parecían suficientemente buenas, y el tiempo era razonable. La red final presenta la estructura que se detalla a continuación y que es (96,19,4,1). Bajando esta cantidad de nodos y capas, los resultados no eran buenos, es decir la red no aprendía. Esta

configuración final de la red se detalla en la *Figura 6.11* La estructura es de 4 capas. La implementación de las capas se detalla a continuación. Todas las sinapsis de la red son del tipo todos a todos. Es decir, cada nodo está conectado con todos los nodos de la capa siguiente. De esta manera podemos captar más relaciones entre los datos.

6.5.2.1. Capa 1- Capa de Entrada

La cantidad de nodos es igual a la sumatoria de todos los valores posibles de los atributos discretos, más 2, correspondientes a los atributos continuos *CuandoCompro y precio.* En nuestro caso ese valor es igual a 96.
La codificación de entrada para los atributos discretos es la siguiente, si el valor del atributo que representa ese nodo se conoce, el valor de entrada es cero, caso contrario el valor será uno. La función de activación es la función escalón. Debido a que solamente capta la presencia o no del atributo. Para los valores continuos la función de activación es la sigmoidea. De la misma manera que lo hicimos en el ejemplo anterior.

6.5.2.2. Capa 2 – Primer Capa Oculta

Esta capa está totalmente conectada a la capa inicial. Consta de 19 nodos. Uno por cada atributo, la idea es que estos capten la importancia de cada uno de los valores de sus atributos, en el caso de los atributos discretos; más algunas posibles relaciones que puedan existir. La función de activación utilizada es la sigmoidea.

6.5.2.3. Capa 3 - Segunda Capa Oculta

Esta capa está totalmente conectada a la segunda capa. La tercer capa consta de 4 neuronas la función de activación es la sigmoidea. La idea de esta capa, es captar las posibles relaciones existentes entre las variables, que escapan a las posibilidades de la capa anterior.

6.5.2.4. Capa 4 - Concepto

Por último la capa 4, que consta de una sola neurona, y es la encargada de decir si el producto es o no es interesante a los gustos y hábitos del cliente. Esta neurona está totalmente conectada a la capa 3, y el objetivo es captar la suma de importancia de las capas anteriores.

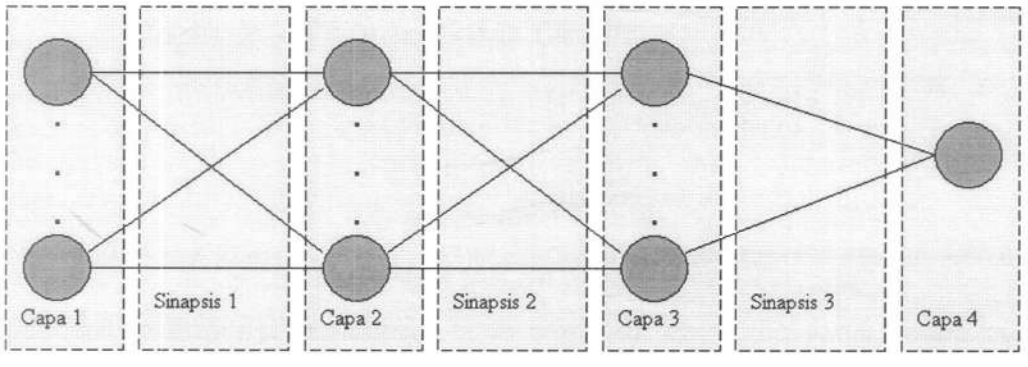

Figura 6.11 - Estructura de la red neuronal para representar perfiles de usuarios

6.6. Comentarios Finales

En este capítulo dimos una pequeña reseña sobre redes neuronales. Explicamos sus principales atributos y dimos una idea de su desempeño en tareas de aprendizaje. También se explica cómo se codifican las variables para realizar el aprendizaje; y cómo se codifican los valores en la red para ser entendidos por la red neuronal.
Para ver cuál fue el resultado obtenido, y los casos de test de esta implementación, es conveniente pasar al capítulo VII, en el cual se realiza un análisis del desempeño de esta implementación del agente.
Queda para algún futuro trabajo implementar distintos tipos de redes neuronales, testearlas y compararlas con otras redes neuronales, y con otras técnicas de aprendizaje.

Capítulo VII

7. Resultados Obtenidos

En este capítulo analizaremos el desempeño de las distintas implementaciones de nuestro agente. Consideraremos relativos al desempeño, factores como, tiempos de respuesta, memoria consumida, adaptación a los cambios en los gustos y hábitos del usuario, y velocidad de aprendizaje.

Vendor a sido testeado durante la asistencia de usuarios hipotéticos de una tienda de comercio electrónico, cuyo rubro es el de artículos deportivos (MUNI deportes S.A.). Un usuario en esta tienda de comercio electrónico, navegará el sitio buscando aquella información que es de su agrado. La función del agente es detectar estos gustos tan pronto como le sea posible, y ofrecer al usuario productos acordes a estos gustos.

Realizamos una cantidad de experimentos usando usuarios simulados de la tienda virtual durante un periodo de tiempo. Decidimos simular los usuarios, debido a que de esta manera, podremos testear ciertas situaciones particulares en las que estamos interesados, y que, con usuarios reales podrían ser pasadas por alto. Además, dado que se trata de varias implementaciones distintas del agente, se hace bastante difícil, en un marco de usuarios reales, lograr que los casos de prueba sean homogéneos para cada implementación. Los usuarios de prueba fueron creados teniendo en cuenta usuarios reales de la tienda; cada usuario representa a un grupo de usuarios existentes en la vida real. Los datos para realizar las simulaciones, se obtuvieron por medio de entrevistas a personas que trabajan en la tienda de artículos deportivos; y a

través de una amplia experiencia del autor en esta área, desempeñando tareas estadísticas y de análisis de datos en esta misma empresa.
Los experimentos llevados a cabo se dividen en tres tipos principales, a saber:
Primero, analizamos cómo el contenido del perfil es refinado, y mejorado con el tiempo, a partir de las observaciones tomadas por el agente. Con este experimento buscamos saber si el agente logra captar los intereses particulares del usuario, especializando sus sugerencias hacia los intereses mostrados por el usuario a través de su interacción con el sistema.
Segundo, analizamos la velocidad de adaptación a los cambios en los intereses del usuario para cada una de nuestras implementaciones del agente. En el contexto en que se desempeña el agente, la velocidad en los cambios de intereses por parte del usuario es alta, debido a que podrían cambiar, potencialmente, con cada compra finalizada. Con este experimento buscamos analizar cómo se comporta el agente ante este dinamismo.
Tercero, analizamos el tiempo y memoria promedio, consumido en las tareas de nuestro agente, particularmente nos centraremos en las tareas relacionadas con el aprendizaje. En este sentido hay dos tareas fundamentales a probar; primero, el tiempo y memoria invertidas en el aprendizaje del perfil del usuario; y segundo, el tiempo y memoria invertido en la realización de ofertas pertinentes al usuario, es decir en seleccionar el producto más adecuado para el cliente, a partir de una cierta cantidad de ofertas posibles.

Estos experimentos se detallan en las siguientes secciones.

7.1. Refinamiento del Perfil de Usuario a lo Largo del Tiempo

En este experimento, analizamos el desempeño, de cada una de las tres implementaciones de *Vendor,* en la tarea de construcción y mejora del perfil del usuario a lo largo del tiempo. Este análisis fue realizado durante la asistencia de nuestro agente a usuarios hipotéticos de una tienda de comercio electrónico. En este análisis se llevan a cabo dos tipos de experimentos. En el primero, el análisis se centrará en el perfil del usuario, su modificación paulatina, y en que punto el agente comienza a dar sugerencias relevantes al usuario.
Para este experimento se le dará al agente cierta cantidad de lecciones, por supuesto, estas lecciones se repiten para cada una de las implementaciones de nuestro agente. Una lección se corresponde, con algunos de los posibles casos que son utilizados por el agente para aprender. En la *Tabla 7.1* Se reiteran las acciones posibles del usuario que brindan una lección, utilizada por el agente para aprender. Es decir, a partir de estas acciones, el agente puede inferir una cierta inclinación en los gustos del usuario, y utilizarlo, o no, para mejorar el perfil que mantiene del usuario.

Lecciones
Cuando un producto aparece en una búsqueda
Cuando un producto es comprado
Cuando se amplía la vista de un producto (Zoom)
Cuando se agrega al carrito
Cuando una oferta no es tenida en cuenta
Cuando es seleccionado un producto, surgido de una búsqueda específica del usuario.
Se toma en cuenta el tiempo que un usuario permanece en una página.
Si la página tiene Scroll, y el usuario lo utiliza, esto también demuestra un cierto interés por lo que está viendo.
Cuando un usuario selecciona el color de su producto. De no haber color de su preferencia esto se utiliza también como una lección para el agente.
Cuando un usuario selecciona el Talle de su producto. Si abandona esta página es muy posible que no haya talles, esto se utiliza también como una lección para el agente.
Cuando el usuario navega utilizando la barra de navegación de la aplicación. Esto le da una relevancia directa a las categorías de productos en los que el usuario está interesado.

Tabla 7.1 - Comparación tiempo consumido

En el segundo experimento, analizaremos el refinamiento del perfil con respecto a algunos conceptos particulares, propios del dominio; como por ejemplo, la relación existente entre el atributo *CuandoCompro* y la *Categoría* del producto. Esta relación decía que, si un usuario compró zapatillas recientemente, difícilmente vuelva a comprar zapatillas en un futuro próximo. Para este experimento se utilizarán también usuarios simulados del sistema; y se realizará una prueba por cada uno de los conceptos más importantes. El detalle de cada una de ellas se dará junto con la explicación de los experimentos más adelante, en la sección 7.1.2.

7.1.1. Análisis de Refinamiento y Mejora del Perfil a lo Largo del Tiempo

A continuación se detallan los experimentos realizados. Se analiza el refinamiento y mejora sobre el tiempo del perfil del usuario, a partir de las acciones realizadas por usuarios simulados, en una tienda de comercio electrónico.
Cada experimento trata con un tipo de usuario especial, con un determinado perfil predefinido, y que se detalla en cada experimento. A su vez, cada experimento muestra un gráfico de los resultados de estos análisis. Cada uno de los gráficos describe el porcentaje de ofertas pertinentes realizadas por el agente, el porcentaje de ofertas no pertinentes, y las diferencias; para cada una de las sesiones realizadas por el usuario. Una oferta es considerada pertinente cuando es acorde al perfil del usuario predefinido. Una sesión es el tiempo que un usuario permanece en un sitio web con su navegador, hasta que cambia de sitio, o cierra el navegador. En una sesión el agente puede recibir muchas lecciones, potencialmente, tantas como acciones realice, esta información está representada en los gráficos de la siguiente manera: Cada sesión viene etiquetada por

un número más una L final, por ejemplo 3L, esto quiere decir que el agente lleva recibidas 3 lecciones.
En todos los experimentos se parte de un perfil sin conocimiento alguno, para todas las técnicas de aprendizaje; es decir, el perfil no tiene conocimiento alguno sobre el usuario, esto se debe a que, precisamente, lo que queremos medir es la capacidad del agente para refinar y mejorar el perfil con las continuas lecciones inferidas.
A continuación se detallan cada uno de los experimentos realizados.

7.1.1.1. Experimento 1

Escenario – El Viajante

El usuario en cuestión, trata de comprar bolsos y accesorios, normalmente utilizados en viajes, a lo largo de las distintas sesiones, busca y analiza ofertas sobre este tipo de productos.

Objetivo

Analizar la adaptación del perfil del usuario. Para esto se medirá cuantas de las ofertas realizadas por el agente coinciden con los gustos del usuario, *el viajante,* definido en la sección anterior.

Experimento

El usuario realizó 4 sesiones, aportando dos, una, dos, y una lección, en cada sesión al agente. La navegación giró en torno a los gustos del usuario, sin llegar a comprar ningún producto. Todas las lecciones son de navegación (utilizando la barra de navegación y los links de los productos)

Resultado

El gráfico de la *Figura 7.1* muestra el resultado de este experimento. Como puede verse, todas las implementaciones mejoraron a lo largo del tiempo; aunque, debido a la poca cantidad de lecciones, la red neuronal no llegó a realizar una suficiente cantidad de ofertas y sugerencias pertinentes, por otro lado, la implementación con algoritmos genéticos logró una excelente performance desde el principio, y la red bayesiana, muestra una rápida adaptación al concepto, a pesar de unas primeras respuestas negativas.

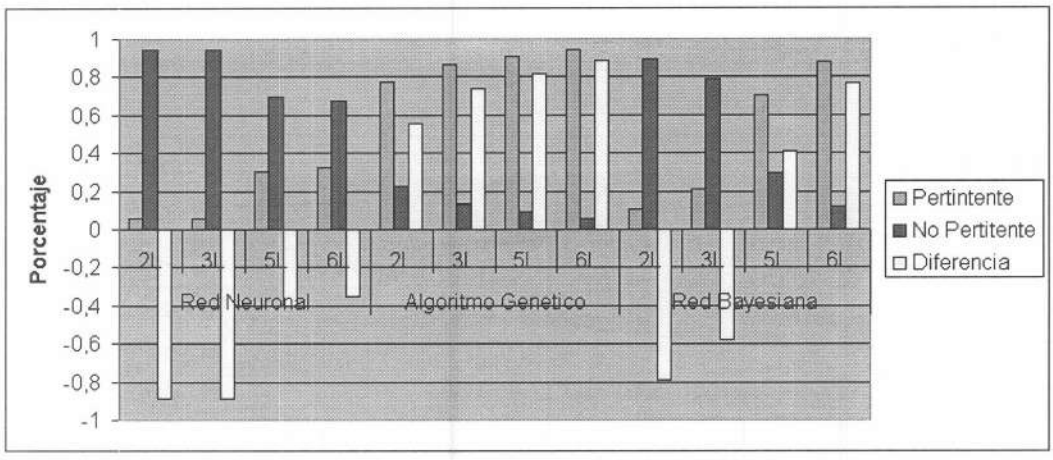

Figura 7.1 - Adaptación del usuario "Viajante"

7.1.1.2. Experimento 2

Escenario – El Futbolista

En este caso, el usuario en cuestión, está interesado en todo producto que tenga algo que ver con el fútbol, a lo largo de las distintas sesiones busca, analiza, y compra ofertas sobre productos relacionados con este deporte.

Objetivo

Analizar la adaptación del perfil del usuario. Para esto se medirá cuantas de las ofertas realizadas por el agente coinciden con los gustos del usuario *el Futbolista,* definido en la sección anterior.

Experimento

El usuario realizó 5 sesiones, aportando dos, cuatro, cuatro, cuatro, y tres lecciones, en cada sesión, respectivamente. La navegación giró en torno a los gustos del usuario hacia los productos de fútbol, comprando algunos productos de distintas categorías.; por ejemplo, *Medias* y *Botines,* las lecciones en esta prueba son más variadas que en el caso anterior e incluye, compra, detalles, y búsqueda de productos.

Resultado

El gráfico de la *Figura 7.2* muestra el resultado de este experimento. Todas las implementaciones mejoraron a lo largo del tiempo. El aprendizaje del perfil por parte de algoritmos genéticos y redes bayesianas, resultó de una muy buena performance, aprendiendo rápidamente el concepto. Redes neuronales, Obtuvieron una performance, más bien, mediocre, mejorando a lo largo del tiempo, pero con una cantidad de ofertas no pertinentes muy elevada.

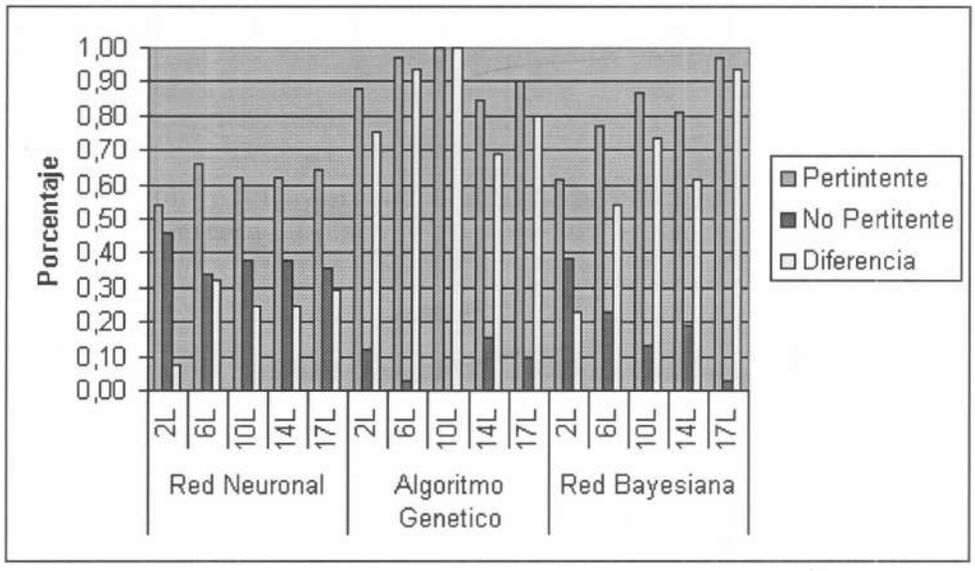

Figura 7.2 - Adaptación del usuario "Futbolista"

7.1.1.3. Experimento 3

Escenario – El Utilero

El usuario en cuestión está interesado en todo producto que tenga algo que ver con accesorios, como ser, medias, bolsos, y artículos de protección al deportista.

Objetivo

Analizar la adaptación del perfil del usuario, para esto se medirá cuantas de las ofertas realizadas por el agente coinciden con los gustos del usuario, *el Utilero,* definido anteriormente.

Experimento

El usuario realizó 4 sesiones, aportando dos, dos, tres, y tres lecciones en cada sesión, respectivamente. La navegación giró en torno a los gustos del usuario hacia los productos de accesorios descriptos anteriormente. Comprando algunos productos de distintas categorías. Por ejemplo Medias y Bolsos, eligiendo siempre los de menor valor Los tipos de lecciones son variadas incluyendo compra de productos.

Resultado

El gráfico de la *Figura 7.3* muestra el resultado de este experimento. En este se ve el rápido aprendizaje de las redes bayesianas y los algoritmos genéticos. Aunque, otra vez, las redes neuronales presentan un rendimiento discreto; esta vez, mostrándose oscilantes en la última sesión, disminuyendo de manera notoria la efectividad de las sugerencias.

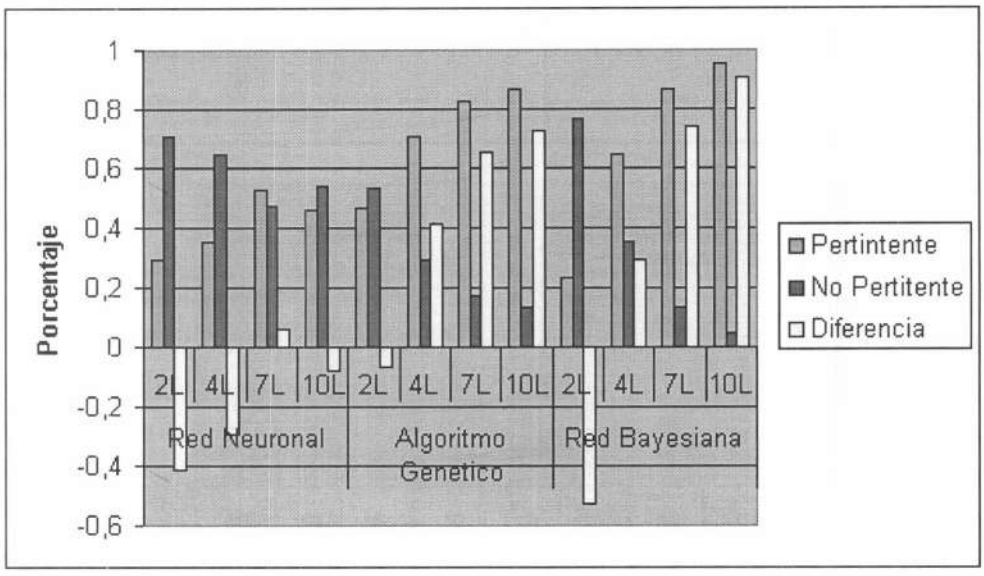

Figura 7.3 - Adaptación del usuario "Utilero"

7.1.1.4. Experimento 4

Escenario – El Ahorrativo

En este caso el usuario, compra, observa, y busca, productos de bajo costo. Sin importar la categoría, lo importante es que sea barato, o sea, no importa la calidad, ni que los productos tengan un tiempo de exposición alto.

Objetivo

Analizar la adaptación del perfil del usuario, para esto se medirá cuantas de las ofertas realizadas por el agente coinciden con los gustos del usuario, *el Ahorrativo* definido en la sección anterior.

Experimento

El usuario realizó 3 sesiones, aportando tres lecciones en cada sesión. Las lecciones giraron en torno a la búsqueda y vista ampliada de los productos más baratos de cada categoría. Todas las lecciones son obtenidas a través de la observación y ampliación de productos, sin llegar a realizar ninguna compra.

Resultado

El gráfico de la *Figura* 7.4 muestra el resultado de este experimento. En este se puede observar una buena adaptación paulatina de todas las implementaciones de nuestro agente, aun así, las respuestas de la implementación con redes neuronales sigue siendo la de menor performance.

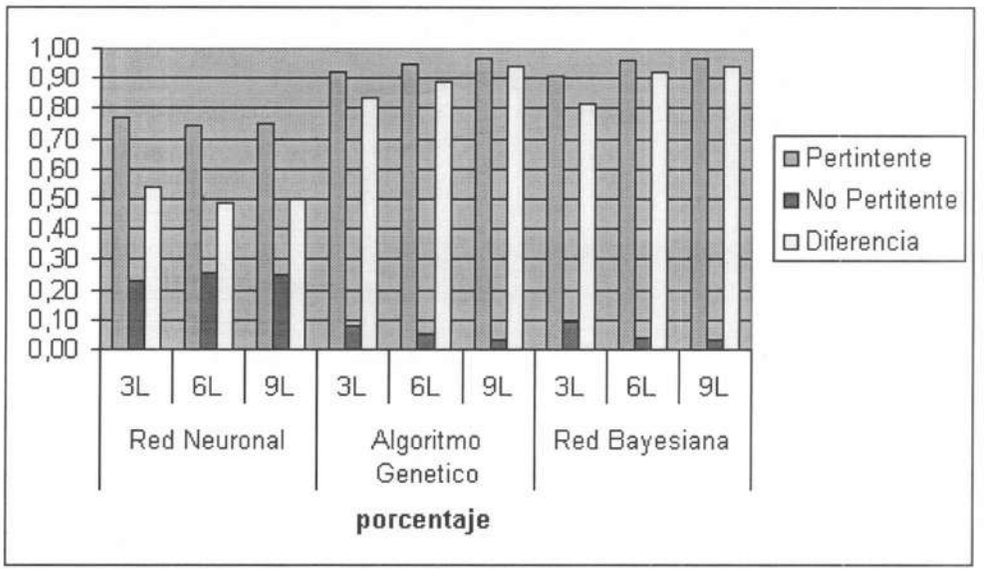

Figura 7.4 - Comprador de ofertas y artículos de segunda

7.1.1.5. Conclusión

Como puede verse a lo largo de estos cuatro experimentos, las implementaciones que utilizan algoritmos genéticos y redes bayesianas resultaron de un muy buen desempeño, luego de una pocas lecciones recibidas logran realizar un alto índice de ofertas pertinentes, es decir acordes al perfil del usuario al que representan. Aun cuándo la cantidad de ofertas no pertinentes dadas por el agente antes de empezar a recibir las lecciones sea muy elevado. Un ejemplo de esto se puede ver en la *Figura* 7.3 correspondiente al experimento número 3, *el utilero*. En este caso se parte en ambas técnicas de un alto índice de respuestas no pertinentes, que son de un 35% para algoritmos genéticos y de un 75% para redes bayesianas, aun así, después de sólo 7 lecciones y tres sesiones, este índice baja a 17% para los algoritmos genéticos, y a 15% para las redes bayesianas.
Por otro lado, el desempeño de las redes neuronales, no fue del todo bueno a lo largo de los cuatro experimentos, puede observarse en los gráficos que, si bien la cantidad de ofertas no pertinentes disminuye a lo largo de las lecciones recibidas, ocurre de manera muy lenta.

7.1.2. Análisis de Refinamiento a lo Largo del Tiempo de Conceptos Particulares

En este segundo experimento analizaremos el refinamiento del perfil con respecto a algunos conceptos particulares, propios del dominio, y que pueden incluir ciertas reglas de negocio particulares; como por ejemplo la relación existente entre el atributo *CuandoCompro* y la *categoría* del producto. Esta relación, decía que si un usuario

compró zapatillas recientemente, difícilmente vuelva a comprar zapatillas en un futuro próximo. Para este experimento se utilizarán usuarios simulados del sistema.
Los experimentos fueron realizados de la siguiente manera, se le dio a cada implementación del agente, lecciones especificas sobre cada tipo de regla o concepto a testear, analizando luego, si las acciones del agente eran acordes a la lección dada.
Aquí no medimos que tan rápido aprende el concepto; medimos, si al cabo de unas pocas lecciones, las cuales se detallarán oportunamente el agente capto o no el concepto.

7.1.2.1. Hábitos Temporales

Escenario

Uno de los conceptos y reglas de negocio más importante son las compras realizadas de manera habitual en determinadas fechas, ejemplos claros de esto son las amas de casa que compran zapatillas para sus hijos luego de cada receso escolar; otro ejemplo podría ser el de una abuela que compra ofertas todas las navidades y días festivos.

Experimento

Para este experimento se realizaron varias pruebas. Estas pruebas se llevaron a cabo simulando usuarios, que realizaban compras, según determinadas fechas, y que estas se repetían cada cierto tiempo.
Utilizamos un usuario, con las características de la *abuela,* descrito anteriormente para testear los hábitos de compra en días festivos. Otro experimento importante, fue probar la rutina de compra, según periodos de tiempo, semanales, mensuales, y anuales. Luego se compararon los resultados obtenidos.

Resultado

Este concepto es aprendido con un alto índice de aceptación por parte de la implementación que utiliza redes bayesianas. Seguido por algoritmos genéticos: algoritmos genéticos, mostraron una buena peformance, a pesar que no hay forma de explicitarle a la técnica que tenga en cuenta rutinas de fechas. Mientras que redes neuronales tienen un desempeño bastante pobre.

7.1.2.2. Relación CuandoCompro – Categoría

Escenario

Existe una fuerte relación entre el atributo *CuandoCompro*, que indica la cantidad de días desde la última vez que un usuario compró un producto de una categoría particular, y el atributo categoría. Una consecuencia obvia de esta relación, es que un usuario no volverá a comprar zapatillas, si compró zapatillas recientemente.

Experimento

Se realizaron dos tipos de pruebas en este experimento. Primero se realizó la compra de un par de botines de fútbol, y luego se le pidió ofertas al agente, para ver si ofertaba productos relacionados con atributos del calzado que compró, pero sin ofertar zapatillas nuevamente. Es decir el resultado esperado es que oferte, por ejemplo, medias de fútbol, o camisetas. En el segundo experimento se realizó la compra de

varios productos, y luego de la compra de cada producto, dejábamos de brindar lecciones relativas a la categoría del producto mostrado. El objetivo es medir la adaptación del perfil a esta regla implícita.

Resultado

El primer caso de prueba fue pedir una oferta inmediatamente después de una compra y ver si ofertaba productos relativos, pero no de la misma categoría, sólo funcionó correctamente con las redes bayesianas, esto se debe a que esta regla está implícita en la concepción de la red bayesiana (remitirse al capítulo IV para más detalles); y sirve para destacar una de las ventajas de esta técnica de aprendizaje por sobre las demás, con respecto a la especificación de reglas del negocio.
El segundo caso de prueba, sólo fue aprendido eficientemente por la implementación con redes bayesianas. Por la misma razón que se especificó más arriba.

7.1.2.3. Relación Marca – Categoría

Escenario

En las búsquedas por categorías, en general, se encuentra implícito el gusto por alguna marca en particular, y viceversa. Es decir, hay una relación entre las marcas, y las categorías de productos. Esta relación es algo sutil, aunque al tenerla en cuenta mejora sensiblemente la performance del agente.

Experimento

Para este experimento, se le brindó al agente varias lecciones en las que se buscaban por categoría, y se seleccionaban ciertas marcas en particular para cada categoría, ignorando las otras. Es decir, se le hacia notar al agente, que el usuario estaba interesado en determinadas categorías, pero de algunas marcas específicas. Luego se analizaron las ofertas realizadas por el agente, para comprobar si estas eran acordes a la categoría y marca seleccionados por el usuario.

Resultado

De igual manera que la relación *CuandoCompro* con *Categoría*, ésta es aprendida por la implementación con redes bayesianas; aunque algoritmos genéticos, en menor medida, dieron resultados aceptables.

7.1.2.4. Análisis de Ruido

Escenario

Tomamos como ruido, aquellas acciones del usuario, que no tienen un objetivo cierto y que puedan hacer que el agente aprenda un perfil erróneo. Un ejemplo sería un usuario curioso que recorre todo el sitio llenando de lecciones al agente, para finalmente comprar algo que no tiene nada que ver con lo que estuvo viendo.

Experimento

En este sentido, se definió un perfil de usuario, de los utilizados en la sección anterior, análisis de refinamiento y mejora del perfil sobre el tiempo, en especial se tomó el

usuario, *el Futbolista*, y se le proveyó al agente lecciones que no tenían que ver con este perfil, sin llegar a comprar nada, y sin relación entre ellos, es decir, ruido.

Resultado

La implementación más robusta a este tipo de ruido fue la red neuronal; debido, principalmente, que requiere gran cantidad de ejemplos para aprender un concepto. Por lo tanto, pequeños "desvíos" del usuario de su objetivo real hacen que el perfil del usuario varié muy poco, mientras que para redes bayesianas el perfil se torna oscilante e impreciso. Algoritmos genéticos, resultó ser un término intermedio entre los dos, oscilando pero en menor medida.

7.1.2.5. Conclusión

En las pruebas sobre el refinamiento del perfil a lo largo el tiempo, testeado con reglas del dominio, y casos particulares, los resultados fueron bastante variados. Por un lado, redes neuronales resultaron robustas al ruido, mientras que redes bayesianas respondió bastante mal, y algoritmos genéticos estuvo en el medio de ambos.
En las demás pruebas sobre relaciones entre atributos y hábitos del usuario, el mejor desempeño se obtuvo utilizando redes bayesianas, pero la implementación con algoritmos genéticos resultó muy buena también; aunque no llego a los niveles de las redes bayesianas.

7.2. Adaptación a los Intereses Dinámicos del Usuario

Como dijimos anteriormente, la adaptación de los gustos dinámicos del usuario es una de las propiedades más interesante de nuestro agente. Los gustos y propiedades del usuario podrían cambiar con cada compra; y el agente debería captar esto con cierta velocidad.
Para realizar esta prueba nos propusimos dos experimentos. Se trata de tomar dos de los usuarios usados para analizar el refinamiento y mejora del perfil sobre el tiempo, cuyos perfiles generados sean más disímiles. Estos usuarios son *el Futbolista* y *el viajante*.
El experimento consiste en darles las mismas lecciones, pero invertidas, es decir las lecciones del *Futbolista* dárselas al *Viajante*, y comprobar si las implementaciones de nuestro agente pueden aprender el concepto fútbol partiendo del perfil del *viajante* y viceversa.
A continuación se detallan estos experimentos

7.2.1. Experimentos Realizados

7.2.1.1. Experimento 1

Escenario – De futbolista a Viajante

Tenemos un usuario simulado, *el Futbolista*, cuyo perfil aprendido por todas las implementaciones de nuestro agente, lo identifica con aquellos productos que tienen

que ver con el fútbol. Pero ahora, este usuario, cambia sus gustos y decide comprar bolsos y accesorios que normalmente se utilizan al realizar un viaje.

Experimento

El experimento consiste en aplicar a los perfiles generados en el experimento de la sección anterior, "*El Futbolista*", las mismas lecciones que se aplicaron en el experimento "*El Viajante*", también realizado en la sesión anterior. Recordemos que en la sesión anterior partíamos de un perfil equiprobable y no sesgado. Ahora partimos de un perfil sesgado hacia todo lo que tenga que ver con fútbol, y queremos ver si puede adaptarse al nuevo concepto, y cuánto tarda en hacerlo.

Resultado:

En la *Figura* 7.5 puede verse el resultado de este experimento. Observando el gráfico podemos concluir que: Todas las implementaciones mejoraron a través de las distintas lecciones; que tanto algoritmos genéticos como redes bayesianas presentan una adaptación al concepto mucho más rápida que redes neuronales, ya que obtienen un porcentaje de ofertas pertinentes muy alto con sólo 6 lecciones recibidas. Redes neuronales tiene una adaptación tan lenta como la mostrada en la sección anterior, análisis de refinamiento y mejora del perfil sobre el tiempo.

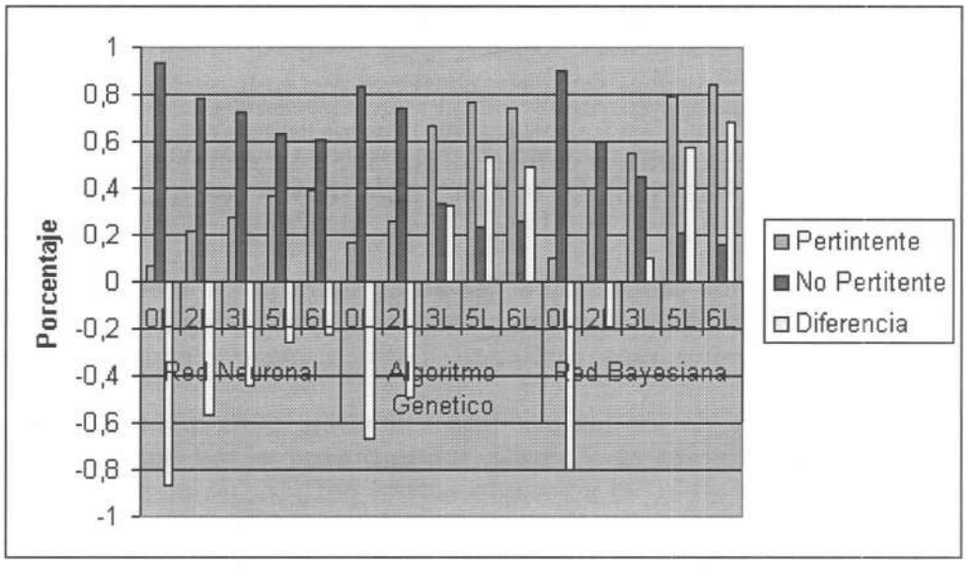

Figura 7.5 - De Futbolista a Viajante

7.2.1.2. Experimento 2

Escenario – De Viajante a Futbolista

De manera inversa al experimento realizado anteriormente, ahora tenemos un usuario cuyo perfil aprendido por todas las implementaciones de nuestro agente, lo identifican con todo aquello que tenga que ver con artículos normalmente utilizados para viajar, por ejemplo, bolsos; pero, ahora, este usuario cambia sus gustos, y decide comprar artículos relacionados con el fútbol.

Experimento

El experimento consiste en aplicar a los perfiles generados en el experimento "*el Viajante*" las mismas lecciones que se aplicaron en el experimento "*el Futbolista*" en la sección anterior. La idea de este experimento es la misma del experimento anterior. Sólo que ahora, tenemos que para el futbolista, utilizamos muchas más lecciones, estas son un total de 17.

Resultado:

En la *Figura 7.6* se encuentra graficado el resultado de nuestro experimento. Observando el gráfico podemos decir que, al igual que en el experimento anterior, la adaptación de la implementación que utiliza redes bayesianas y la que utiliza algoritmos genéticos, resultaron muchos mejores que la que utiliza redes neuronales, llegando a lo largo de las lecciones a obtener altos porcentajes de ofertas pertinentes, 90% para redes bayesianas y 80 % para algoritmos genéticos, según puede verse en el gráfico de la *Figura* 7.6.

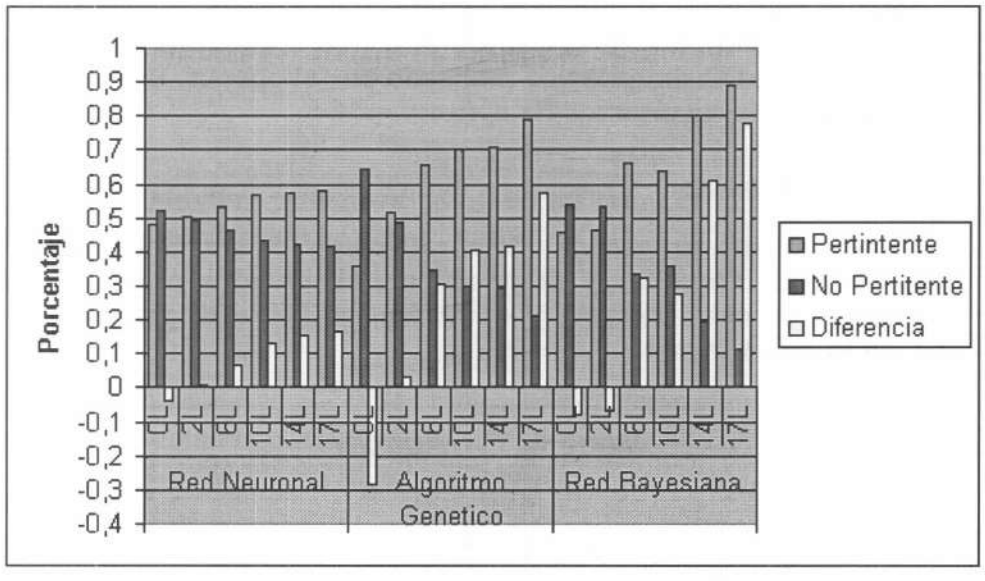

Figura 7.6 - De Viajante a Futbolista

7.3. Análisis de Tiempo y Memoria

Como último punto a analizar en las distintas implementaciones de nuestro agente; evaluaremos el desempeño de estos en los referente a memoria y tiempos consumidos en sus tareas cotidianas.
Recordemos que el agente realiza varias tareas por si mismo, entre ellas, interactúa con el usuario, interactúa con la aplicación, mantiene y mejora el perfil del usuario al que asiste, etcétera. En este sentido las tareas criticas, y que más tiempo y memoria consumen, son las relativas al aprendizaje y mantenimiento del perfil. Estas tareas pueden dividirse en dos grandes conjuntos, aquéllas que involucran aprender a través de las lecciones indirectas, inferidas del uso de la aplicación por parte del usuario, las que en adelante llamaremos aprendizaje; y las tareas que requieren del agente utilizar

el conocimiento ya aprendido para, por ejemplo, diagnosticar la importancia relativa de un producto para el usuario al que asiste, y que llamaremos diagnóstico.

7.3.1.1. Escenario

Para este experimento, se utilizará la información obtenida a partir de los experimentos realizados en las secciones anteriores.
Las pruebas Fueron hechas utilizando una máquina Pentium II con un procesador de 333 mz, y 128 MB de memoria.

7.3.1.2. Experimentos

Los experimentos realizados son los que se describieron en las secciones previas, para obtener los tiempos y memoria aproximados se recurrió al framework implementado, para llevar a cabo el desarrollo de nuestro agente, y que se describe en al apéndice B. Éste realizó el almacenamiento de los datos relativos a tiempos y memoria de cada lección aprendida, y de cada diagnóstico realizado por el agente. Luego tomamos estos datos los promediamos y presentamos de manera gráfica.

7.3.1.3. Resultados

En la *Figura* 7.7 se muestra un gráfico comparativo con los resultado promedios de los tiempos consumidos en aprender y diagnosticar, observados para cada una de las implementaciones de nuestro agente; y en la *Figura* 7.8 la el promedio de memoria consumida en las mismas tareas.

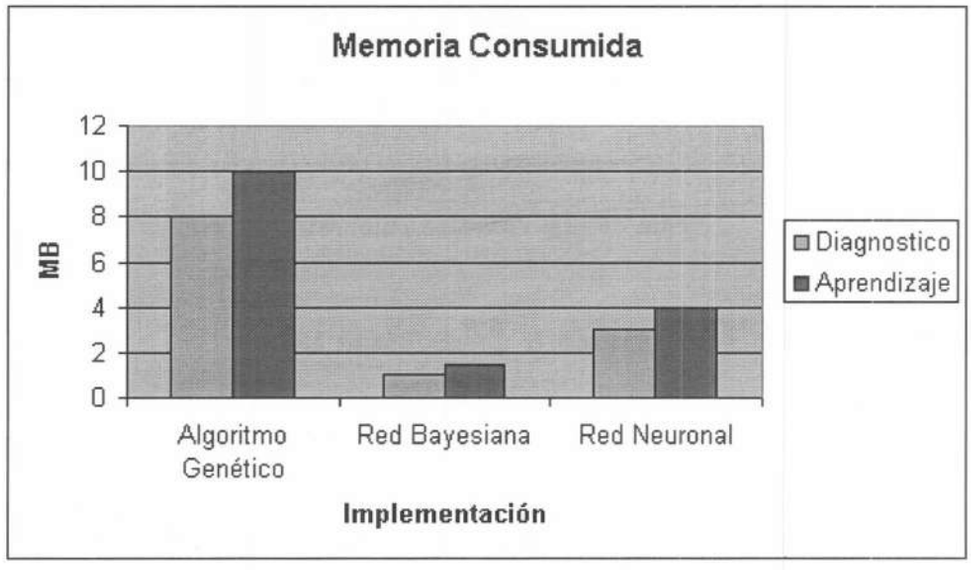

Figura 7.7- Comparación memoria consumida

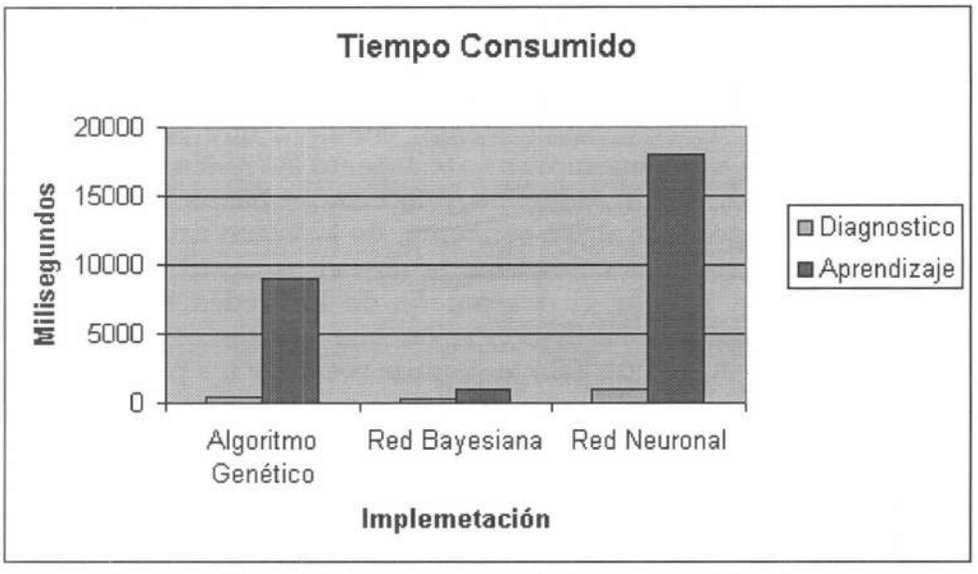

Figura 7.8 - Comparación tiempo consumido

Como puede observarse en las figuras anteriores, en términos de tiempo de respuesta y memoria, resultó ser la red bayesiana, con la implementación de nuestro algoritmo de adaptación de probabilidades, la más eficiente (alrededor de 250 milisegundos). Algoritmos genéticos resultaron eficientes para el diagnóstico (300 milisegundos), aunque se mostraron mucho más lentos en la tarea de aprendizaje (hasta 10 segundos), y el consumo de memoria resultó ser el más alto (alrededor de 9 megabytes en promedio para diagnóstico y aprendizaje. Redes neuronales resultaron de muy mal rendimiento en todos los aspectos, alcanzando el punto más bajo en la tarea de aprendizaje, donde llego a tardar hasta 30 segundos en una lección.
En resumen, considerando una relación entre tiempo y memoria, resulta mejor la implementación de redes bayesianas con un algoritmo adaptativo de probabilidades, segundo La implementación de algoritmos genéticos con memoria, finalmente la red neuronal implementada como un perceptrón multicapa utilizando backpropagation para el aprendizaje.

7.4. Comentarios Finales

A lo largo de las pruebas realizadas podemos ver que la implementación de nuestro agente que utilizó la técnica de redes bayesianas, resultó mejor que las demás implementaciones. Primeramente, debemos aclarar que el algoritmo implementado para el aprendizaje hizo que la red fuera mucho más rápida en la tarea de recalcular los valores de las probabilidades, como así también disminuyó la cantidad de memoria utilizada para tal proceso.
Además, redes bayesianas resultan buenas para modelar el conocimiento de fondo y reglas de negocio comunes, que con otras técnicas, resulta difícil de realizar. La eficiencia de las redes bayesianas se vio demostrada a lo largo de la mayoría de los experimentos, siendo superado sólo en uno, por la implementación que utiliza algoritmos genéticos con nuestra implementación de memoria. Por otro lado, la

implementación que utiliza redes neuronales, resultó bastante pobre e inapropiada para el dominio en que se desempeña el agente; debido a que tiempos de respuesta rápidos son importantes en este dominio; en este aspecto las redes neuronales tuvieron tiempos excesivamente altos, y aun, solucionando este problema, la gran cantidad de ejemplos necesarios para aprender algún concepto, no lo hacen una opción tentadora.
Por otro lado, algoritmos genéticos con memoria dieron un resultado bastante bueno llegando en varios casos a igualar el desempeño de las redes bayesianas. En este sentido tiene vital importancia la implementación que utilizamos, así como también, utilizar la noción de memoria, ya que esto le permite recordar los pasos anteriores para poder clasificar e inferir el siguiente paso. Creemos que sería válido seguir probando con esta implementación, que incluso, podría llegar a ser superior en otros dominios donde las reglas del negocio no estén definidas claramente.
A continuación en la *Figura 7.9* se muestra un cuadro comparativo de los resultados obtenidos en cada uno de los experimentos realizados. En este figuran todos los experimentos en un eje, y las implementaciones de los algoritmos en el otro, coloreando la intersección de los ejes, en el caso que esa implementación haya sido la de mejor performance en ese experimento. En algunos casos los resultados fueron bastantes parecidos, sobre todo entre la red bayesiana y el algoritmo genético, en esos casos decidimos declarar un "empate", y en el gráfico figuran coloreados ambos.

		Red Bayesiana	Red Neuronal	Algortimo Genetico
Tiempo	de Aprendizaje			
	De Inferencia			
Memoria Consumida	Al inferir			
	Al Aprender			
Aprendizaje	Experimento 1			
	Experimento 2			
	Experimento 3			
	Experimento 4			
Adaptacion	Experimento 1			
	Experimento 2			
Conceptos Particulares	CuandoCompro			
	Ruido			
	Marca Categoria			
	Habitos Teporales			

Figura 7.9 - Cuadro comparativo

Capítulo VIII

8. Cometarios Finales

En este capítulo final expresaremos una serie de conclusiones acerca del trabajo realizado, se dará un resumen del mismo, y destacaremos las razones por las cuales las propuestas realizadas sirven para direccionar, algunos tipos de problemas, de mejor manera que otras posibles soluciones existentes. También analizaremos los posibles pasos a seguir, y cuáles podrían ser algunas mejoras al presente trabajo.

8.1. Resumen

En el presente trabajo se ha analizado, implementado, y probado, tres técnicas de aprendizaje de máquina, todas estas técnicas se caracterizan por ser robustas, complejas, y ampliamente probadas y utilizadas, en el campo de la inteligencia artificial. Técnicas utilizadas aquí con un nuevo enfoque, el de resolver problemas de sobrecarga de información, y asistencia personalizada de usuarios. Este enfoque tiene especial interés en el área de agentes de interfaz autónomos.
Los Agentes de Interfaz Autónomos [AIA/1997] asisten a usuarios en la realización de tareas cotidianas, intentando actuar como un secretario humano lo haría. Para ello debe conocer al humano al que asiste, pudiendo, de esta manera, cumplir su tarea en

forma eficiente. Para conocer al usuario, el agente deberá aprender los gustos de este. En nuestro caso el aprendizaje se realiza, netamente, a través de la observación del uso del sistema por parte del usuario, sin consultas del agente, ni respuestas directas del usuario.
El entorno en que se desempeña el agente es un sitio de comercio electrónico; algunas de sus tareas son, observar los hábitos, y acciones del usuario, tomando esta información dinámicamente. La información recogida, será utilizada para inferir cuáles son las acciones que el usuario seguirá en su recorrido por el sitio. Este tipo de análisis y colecta de datos dinámicos, posee una función objetivo difusa, y en los que, en general, no se cuenta con datos previos para realizar el entrenamiento. Este factor hace que técnicas de data mining y machine learning clásicas, deban ser reconsideradas, fallen, o quizás sean descartadas.
En particular, un problema importante, es resolver cómo manejar la amplia cantidad de datos que se pueden tomar de manera directa, a partir de las acciones del usuario, aun de manera fuera de línea[KOH/2001]. Este problema se ve ampliado si queremos tomar estos datos dinámicamente (en linea), procesarlos, y darle una respuesta al usuario de manera online, este tipo de tarea es específico y de suma importancia en el campo de Agentes de Interfaz. El presente trabajo brinda las siguientes contribuciones para lidiar con el problema planteado.
Como ya dijimos, tres técnicas de aprendizaje fueron utilizadas para implementar agentes de interfaz; estas fueron, redes neuronales, redes bayesianas, y algoritmos genéticos, todas seleccionadas por sus características evolutivas, lo que permite que el agente aprenda paulatinamente, con cada lección o acción aportada por el usuario. Con respecto a cada una de las técnicas se lograron los siguiente objetivos.

Redes Bayesianas:

Sin dudas, los resultados más interesantes los obtuvimos con esta implementación. Sobre todas las cosas, se destacaron su eficiencia, y el consumo de memoria, muy inferior a las otras técnicas; además, la capacidad de aprendizaje y adaptación resultaron excelentes.
Para lograr esto debimos modificar la técnica utilizada de manera clásica, y recurrir a un enfoque un tanto más evolutivo; evitando también, tener una fuente de almacenamiento, para las probabilidades condicionales de la red. Por otro lado, mucho de la eficiencia se debe a que el algoritmo utilizado fue implementado por nosotros. El algoritmo es, sobre todas las cosas, sencillo y eficiente. A pesar de la sencillez del algoritmo, esta implementación fue mucho mejor que las de redes neuronales.
Redes bayesianas demostraron también que, con la intervención de un experto de dominio, pueden mantener una gran cantidad de conocimiento de las reglas del negocio", factor que puede marcar la diferencia con respecto a otras técnicas. Además, una red bayesiana no solamente infiere un resultado, sino que brinda una explicación o justificación de este resultado, mostrando cómo influyen las demás variables en el resultado final, lo que puede ser de mucha importancia para analizar el desempeño de la red.

Algoritmos Genéticos:

La segunda técnica implementada, es la de algoritmos genéticos, en este caso se recurrió a la implementación de muchas configuraciones, variando significativamente, unas de otras. El grado de estas implementaciones va, desde la más sencilla, hasta la más compleja.

Se modificó el framework de algoritmos genéticos, y se utilizó el concepto de memoria para algoritmos genéticos, empleado en trabajos anteriores, para resolver problemas como los del viajante de comercio. Esta implementación resultó de un muy buen desempeño; logrando, en algunos casos, resultados tan buenos o mejores, que las redes bayesianas. Además, algoritmos genéticos, demostraron una alta capacidad de búsqueda y aporte de soluciones innovadoras; debido, fundamentalmente, a la mutación de individuos. Esto hace que cambios paulatinos, o bruscos del perfil sean, igualmente aprendidos con facilidad.
Algoritmos genéticos demostraron resultados muy satisfactorios, tanto en el aprendizaje, como en la adaptación a los cambios de gustos y hábitos del usuario.
Los problemas más grandes encontrados en la implementación de esta técnica son, el consumo de memoria, y el tiempo de aprendizaje.

Redes Neuronales

La implementación utilizando redes neuronales resultó un tanto inadecuada por varios motivos. El principal, es el tiempo de aprendizaje; luego, el consumo de memoria, exageradamente alto; y por ultimo, pero no menos importante, su capacidad de adaptación y aprendizaje, fueron muy inferiores a los obtenidos por las otras implementaciones. Como aporte positivo, demostró una robustez muy amplia al ruido.

Sobre la base de los resultados aquí obtenidos, y a modo de resumen, podemos decir que:
La implementación de agentes utilizando una red bayesiana resultó excelente para este dominio, en el que conocemos las variables, y tenemos un amplio conocimiento del dominio, el cual nos permite identificar cómo influyen unas sobre otras. En estos casos redes bayesianas es la mejor opción.
La implementación de algoritmos genéticos utilizando memoria, resulta excelente en los siguiente casos. Cuándo los conocimientos del dominio son escasos. Cuándo se desee una búsqueda de soluciones más profunda. Cuando se requieran propuestas nuevas y variedad en la población de perfiles, logrando esto mediante la utilización de cruza y mutación.
Redes neuronales resultaron una herramienta a implementar en los casos en que la adaptación pueda ser lenta, y deba ser robusta al ruido, y difícil de perturbar.

Framework

Para realizar el análisis de las técnicas, hemos recurrido a la implementación de un framework, el cual facilita tal tarea. Este framework permite configurar el dominio en el cual se desempeñará el agente, permite seleccionar entre una o varias técnicas de aprendizaje a la vez; además, permite que muchos agentes, los cuales pueden implementar distintas técnicas de aprendizaje, aprendan de las lecciones del usuario. Así como también, realizar los casos de prueba más sencillos, pudiendo ejecutar las mismas tareas con muchos agentes a la vez. Almacenando los resultados del aprendizaje y diagnostico para cada uno de ellos.

8.2. Trabajo Futuro

Como extensión al trabajo realizado en esta tesis se incluye, por un lado, el tratamiento y análisis de nuevos dominios de comercio electrónico; por otro lado, el análisis de nuevas técnicas, y la modificación de las técnicas aquí implementadas; y finalmente, la ampliación y modificación del framework desarrollado.
Existen varios trabajos propuestos para el futuro, los cuales, detallaremòs a continuación.

Modificación de las técnicas implementadas

Existen varias modificaciones interesantes para realizar y analizar. Entre ellas, figura la modificación del algoritmo de adaptación implementado para redes bayesianas, haciendo que sea variable a lo largo de la cantidad de lecciones recibidas, o incluso, que varié según la cantidad de atributos involucrados. Con esta modificación lograríamos disminuir el sesgo de la red hacia los productos muy visitados, logrando una adaptación mayor a los cambios en los gustos del usuario.
La implementación de algoritmos genéticos puede modificarse para que sea mucho más eficiente. Pueden explorarse también nuevas configuraciones, y nuevas funciones de fitness para la clasificación de los individuos. Analizar la combinación de algoritmos genéticos con otros tipos de técnicas, es una opción a tener en cuenta; en este caso, el algoritmo genético, provee búsqueda y adaptación, y la otra técnica deberá proveer, reglas básicas y conocimiento del dominio.

Implementación de nuevas técnicas

Por el lado de las redes bayesianas, y algoritmos genéticos; el trabajo realizado, fue amplio. Ahora, en lo que a redes neuronales respecta, la cantidad posible de configuraciones, y su amplia variedad, impiden que el trabajo sea exhaustivo, o permita descartar a las redes neuronales para este tipo de dominio. Por otro lado, trataremos en el futuro de realizar un análisis sobre el desempeño de distintas topologías de redes neuronales, para la generación de perfiles dinámicos de comercio electrónico.
Otro factor interesante, sería analizar la conducta del usuario en el entorno de comercio electrónico, como una serie de tiempo; que permita, en base a la memorización de algunas acciones pasadas predecir las acciones futuras. Esto, combinado con el enfoque actual, podría ser más productivo y mejorar de manera significativa la performance.
Finalmente, el análisis de otras técnicas complejas, o mixtas, en este tipo de dominios; sería muy útil, y brindaría un marco mucho más claro, para quienes deban realizar algún tipo de implementación similar.

Testeo de nuevos dominios

Analizar otros dominios de comercio electrónico, de preferencia, ampliamente distintos del utilizado en esta tesis; pero que respete la idea inicial de estar enmarcados en el ámbito del comercio electrónico, y que respeten el modo de aprendizaje mediante la sola observación. Sería sumamente útil para quienes tengan que implementar técnicas de aprendizaje en dominios similares.

Modificación del Framework

La modificación y ampliación del framework permitirá testear nuevos dominios, y nuevas técnicas de aprendizaje, de manera muchos más sencilla. Una ampliación interesante para hacerle, es que soporte el almacenamiento, y re-ejecución automática de los tests, de manera off line, simulando a un usuario.

Verificación del desempeño en entornos reales

La implementación de alguna de estas técnicas en un ambiente real, para verificar su desempeño, sin duda, brindaría un claro aporte al área.

Nuevas Técnicas

Finalmente, sería interesante proponer algún tipo de técnica genérica para este tipo de problemas. Hay, sobre este punto en particular, muchas posibilidades a explorar. Pero seguramente, una de las más productiva, sería alguna que convine el conocimiento de los hábitos temporales del usuario, por ejemplo las fechas importantes, con alguna técnica sencilla, que convine el conocimiento de las últimas transacciones realizadas, e infiera posibles pasos a seguir.
El estudio de la combinación de técnicas difusas, con técnicas matemáticas y de data mining, para análisis y aprendizaje de los dominios, sería una opción a analizar, también muy interesante.

Apéndice A

A. Análisis Estadístico de los Datos

En este apéndice se presenta un breve análisis estadístico de los datos provistos por la empresa MUNI deportes SA. Estos datos corresponden a los movimientos de compraventa de la empresa entre los años 1990 y 1998.
El fin de este apéndice es entender algunos de los diseños y, decisiones en general, por los que optamos; obteniendo, de esta manera, un conocimiento a priori del dominio. Además, buscamos captar la relevancia presunta de cada atributo en juego, y posibles reglas de negocios que sean útiles, e interesantes, para un análisis detallado de la performance de los algoritmos.

A.1. Conceptos Teóricos

Presentamos algunos conocimientos teóricos mínimos, utilizados comúnmente en aprendizaje de máquina y data mining; para una mejor comprensión del análisis a desarrollar.

A.1.1. Dominio

Nos referimos al dominio, como la caracterización de todas las variables que involucra la tarea a aprender. Como ejemplo, en nuestro caso, trataremos con una función que va, desde una cierta cantidad de atributos, a algún valor real representando la importancia del producto para el usuario.

De manera Formal:

Sea:

$$D = \{ \langle x, f(x) \rangle \ / \ x \in V_1 \times V_2 \times \ldots \times V_n \}$$

el conjunto de instancias de entrenamiento, donde:

V_i es el conjunto de valores posibles del i-ésimo atributo de las instancias

$f: V_1 \times V_2 \times \ldots \times V_n \rightarrow C$ es la función de clasificación

C es el conjunto de clasificaciones posibles de las instancias.
Deseamos encontrar una hipótesis que.

$$(\forall \langle x_i, f(x_i) \rangle \in D) \ (h \wedge x_i) \ |\!\!- f(x_i)$$

En nuestro caso encontramos que la f a resolver es la siguiente. Dada una cantidad de atributos referentes a artículos deportivos y fechas. Tratar de predecir el grado de interés despertado en el usuario.
Por lo tanto C es real, y la podemos definir en el intervalo [0,1]. Con respecto al dominio de nuestra función; podemos ver en la *Tabla A.1* un breve resumen de cada uno de los atributos que lo conforman.

A.1.2. Datos Desconocidos

Se denomina valor desconocido, a un determinado valor de una categoría, para el cual no se conoce su valor; por ejemplo, sabemos que un cliente compró zapatillas pero no sabemos de que color.
La importancia y efecto de los valores desconocidos es de gran impacto para algunas técnicas de aprendizaje, pudiendo causar el fracaso de estas.
En nuestro caso particular, podemos decir que no tenemos valores desconocidos, ya que sabemos cada atributo del producto que el usuario está viendo en cada momento, así como también la fecha y hora, día, etcétera.

A.1.3. Ruido

Se denomina ruido a valores mal clasificados en una muestra de datos, por ejemplo, si el cliente estuviera comprando medias rozas; pero, por un error de carga, se ingresa al sistema que compró medias azules. Factores de ruido en la muestra son muy comunes, y al igual que los valores desconocidos, suelen tener un impacto muy profundo en las distintas técnicas de aprendizaje; sobre todo, porque puede hacer que se infieran conclusiones erróneas.
En nuestro caso podemos obviar el ruido en la muestra, debido a que aprenderemos sobre la base de la observación directa del usuario; y en todo caso, el ruido que pueda llegar a ocurrir es producido por el usuario. Esto ocurre, por ejemplo, al realizar búsquedas que no tienen que ver con sus intereses, de esta manera hará que el agente aprenda lecciones erróneas; pero este ruido es producido por la operación, y no por la mala catalogación de los datos.

A.2. Caracterización del Dominio

El dominio a analizar es el de una aplicación de comercio electrónico, como ya dijimos anteriormente. Pero particularmente, nos referimos a la venta de artículos deportivos, el dominio es el de una tienda de deportes virtual. En donde se venden, por ejemplo, pelotas, raquetas buzos, camisetas de fútbol Etcétera. Los datos analizados corresponden a la empresa Muni Deportes S.A., estos fueron recopilados entres los años 90 y 98, con una marcada tendencia alcista en las ventas, debido a que durante este periodo la empresa creció a razón de 2 sucursales por año, contando en el año 98 con 28 bocas de venta.

A.2.1. Nuestro Dominio

Ahora analizaremos con detalle cada atributo, y cuáles son las posibles relaciones existentes entre ellos, la caracterización frente al ruido, la existencia de outliers, etcétera. Para poder analizar la complejidad de la tarea a realizar. Este análisis será solamente una primera aproximación estadística a los datos.

Nombre	Descripción	tipo dominio
Categorías	Categoría del producto	Discreto
Accesorios	Categoría accesorios	Discreto
Calzado	Categoría calzados	Discreto
Deportes	categoría deportes	Discreto
Entrenamiento	categoría entrenamiento	Discreto
Indumentaria	categoría indumentaria	Discreto
Marcas	categoría marcas	Discreto
CategoríaComprador	a quién está destinado el producto	Discreto
Catprecio	categoría del precio	Discreto
Exposición	tiempo de exposición	Discreto
Precio	Precio del Productos	continuo
Fecha hora	Fecha hora	Continuo
Color	Colores	Discreto
rankingInterno	categorización del desempeño del producto dentro de la empresa	Discreto
CuandoCompro	tiempo transcurrido desde la compra	Continuo
Calidad	calidad del producto	Discreto
Talle	talle del producto	Discreto
Interés	concepto objetivo interés dado el producto	Continuo

Tabla A.1 - Distribución de valores del concepto objetivo.

Como podemos ver, 4 de los atributos son continuos, mientras que los otros 14, son discretos, también podemos ver que el rango de los atributos discretos es, en general, pequeño; esto podría ayudar en la tarea de análisis previo y estadístico, ya que de existir correlaciones, podríamos encontrarlas más fácilmente que si fueran numerosos.

A.3. Características generales del concepto a aprender

El concepto a aprender, es el de una función continua que representa el interés generado por el producto.
Concepto objetivo, La principal dificultad en este dominio es que no conocemos la función, a representar, sólo sabemos que ésta trata de muchos atributos y categorías como ya se describió anteriormente; y no contamos con un tutor que puede decirnos si lo que se infirió está realmente bien. Es éste quizás el factor más interesante de la tarea de generar los perfiles.

A.3.1. Relevancia presunta de cada atributo

Para medir la relevancia se realizó un trabajo estadístico, en base a los datos de ventas generales en una tienda de deportes, Todos los datos que figuran aquí son reales, y corresponden las ventas realizadas por Muni Deportes S.A., entre los años de 1990 y 1998. En este caso se separó (discretizó) el atributo continuo fecha en tres atributos discretos; correspondientes a, Mes del Año, Semana del Mes, y Día de la Semana; para poder analizar de mejor manera el comportamiento temporal de las ventas.

<table>
<tr><th>Nombre</th><th>Descripción</th><th>tipo dominio</th></tr>
<tr><td>Categorías</td><td>
<table>
<tr><th>Rango</th><th>Promedio</th></tr>
<tr><td>Accesorios</td><td>16,44</td></tr>
<tr><td>Calzado</td><td>53,86</td></tr>
<tr><td>Entrenamiento</td><td>14,76</td></tr>
<tr><td>Indumentaria</td><td>14,93</td></tr>
<tr><td></td><td>100,00</td></tr>
</table>
</td><td>Discreto</td></tr>
<tr><td>Accesorios</td><td>
<table>
<tr><th>Rango</th><th>Porcentaje</th></tr>
<tr><td>Bates</td><td>0,55</td></tr>
<tr><td>Bolsos</td><td>27,54</td></tr>
<tr><td>Guantes</td><td>17,43</td></tr>
<tr><td>Paletas</td><td>6,32</td></tr>
<tr><td>Pelotas</td><td>24,30</td></tr>
<tr><td>Protectores</td><td>17,43</td></tr>
<tr><td>Raquetas</td><td>6,43</td></tr>
<tr><td>Total</td><td>100,00</td></tr>
</table>
</td><td>Discreto</td></tr>
</table>

Calzado	Rango: Porcentaje Casual: 15,00 Fútbol: 29,55 Náuticos: 7,50 Running: 14,30 Sandalias: 7,70 Skateboarding: 3,25 Training: 12,40 Tenis: 10,30 Total: 100,00	Discreto
Deportes	Rango: Porcentaje Casual: 11,21 Fútbol: 25,43 Náuticos: 8,65 Running: 14,65 Sandalias: 8,50 Skateboarding: 9,32 Tennis: 12,43 Training: 9,81 Total: 100,00	Discreto
Entrenamiento	Rango: Porcentaje Bicicletas: 13,55 Cintas: 9,00 Relojes: 49,65 Musculación: 27,80 Total: 100,00	Discreto
Indumentaria	Rango: Porcentaje Calzas: 17,54 Camisetas: 21,99 Chombas: 15,32 Joggins: 13,43 Medias: 9,54 Musculosas: 3,43 Remeras: 9,32 Shorts: 9,43 Total: 100,00	Discreto

Marcas — Discreto

Rango	Porcentaje
Adidas	10,00
Asics	5,00
Easton	1,00
Franklin	2,00
Gilbert	1,00
Head	5,44
L.A Gear	11,34
Le Coq Sportif	3,40
Nike	31,88
Penalty	7,50
Reef	6,55
Signia	4,33
Spalding	0,80
Topper	8,98
Vairo	0,78
Total	100,00

CategoriaComprador — Discreto

Rango	Porcentaje
Niño	8,65
Dama	20,54
Hombre	41,54
Adolescente	29,27
Total	100,00

Precio — Continuo

Rango	Porcentaje
de 1 a 30	5,00
de 31 a 100	80,00
de 101 a 170	7,76
de 171 a 240	3,87
de 241 a 500	2,12
de 500 a 700	1,25
Total	100,00

Catprecio — Discreto

Rango	Porcentaje
Oferta	7,60
Liquidación	5,00
Normal	83,77
Sobreprecio	3,63
Total	100,00

Exposición — Discreto

Rango	Porcentaje
Un Mes	12,84
Dos Meses	29,54
Tres Meses	27,43
Seis Meses	15,54
Un Año	14,65
Total	100,00

<table>
<tr><td>Día</td><td><table><tr><th>Rango</th><th>Porcentaje</th></tr><tr><td>Lunes</td><td>4,43</td></tr><tr><td>Martes</td><td>6,44</td></tr><tr><td>Miércoles</td><td>8,70</td></tr><tr><td>Jueves</td><td>8,55</td></tr><tr><td>Viernes</td><td>6,43</td></tr><tr><td>Sábados</td><td>59,65</td></tr><tr><td>Domingos</td><td>5,80</td></tr><tr><td>Total</td><td>100,00</td></tr></table></td><td>Discreto</td></tr>
<tr><td>Semana</td><td><table><tr><th>Rango</th><th>Porcentaje</th></tr><tr><td>Primera</td><td>80,00</td></tr><tr><td>Segunda</td><td>10,00</td></tr><tr><td>Tercera</td><td>5,00</td></tr><tr><td>Cuarta-Quinta</td><td>5,00</td></tr><tr><td>Total</td><td>100,00</td></tr></table></td><td>Discreto</td></tr>
<tr><td>Mes</td><td><table><tr><th>Rango</th><th>Porcentaje</th></tr><tr><td>Enero</td><td>4,33</td></tr><tr><td>Febrero</td><td>3,77</td></tr><tr><td>Marzo</td><td>10,20</td></tr><tr><td>Abril</td><td>6,40</td></tr><tr><td>Mayo</td><td>5,80</td></tr><tr><td>Junio</td><td>8,90</td></tr><tr><td>Julio</td><td>7,80</td></tr><tr><td>Agosto</td><td>10,20</td></tr><tr><td>Septiembre</td><td>8,40</td></tr><tr><td>Octubre</td><td>10,20</td></tr><tr><td>Noviembre</td><td>9,00</td></tr><tr><td>Diciembre</td><td>15,00</td></tr><tr><td>Total</td><td>100,00</td></tr></table></td><td>Discreto</td></tr>
<tr><td>Color</td><td>No disponible</td><td>Discreto</td></tr>
<tr><td>RankingInterno</td><td>No disponible</td><td>Discreto</td></tr>
<tr><td>CuandoCompro</td><td>No disponible</td><td>Continuo</td></tr>
<tr><td>Calidad</td><td><table><tr><th>Rango</th><th>Porcentaje</th></tr><tr><td>Primera</td><td>90,00</td></tr><tr><td>Segunda</td><td>2,60</td></tr><tr><td>Saldo</td><td>7,40</td></tr><tr><td>Total</td><td>100,00</td></tr></table></td><td>Discreto</td></tr>
</table>

Talle	No disponible	Discreto

Tabla A.2 - Distribución y valores de los atributos del dominio

A.3.1.1. Conclusión

En la tabla anterior puede verse la distribución de las ventas según cada atributo. Algunos datos importantes observados para el promedio de las compras hechas son:

1. El 60 % de las ventas se realizan los sábados. Este dato no necesariamente debe ser cierto en el caso de venta a través de Internet.
2. Nike es la marca más vendida. Esto se debe una cuestión de exposición en vidriera del producto.
3. De los deportes, fútbol es el más importante, y tiene influencia directa sobre las compras del usuario.
4. El calzado representa el volumen de ventas más importantes.

En base al análisis realizado, y la opinión de los expertos de la empresa; pudimos obtener los resultados que se expresan en la *Figura A.1;* y que dan una idea de la importancia de los atributos más relevantes.

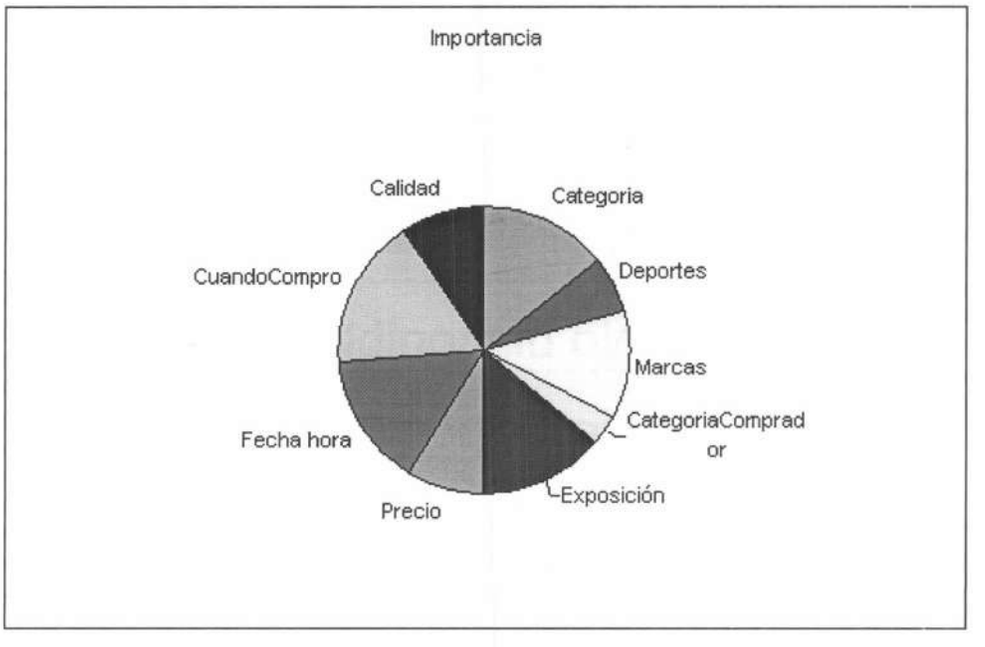

Figura A.1 - Distribución de los Atributos

A.3.1.2. Correlaciones

Hay un problema clave en los datos, y es que nunca en la empresa se tuvo en cuenta la existencia del cliente. Esto se deba a que muchos sistemas no tienen en cuenta a los usuarios como parte del sistema, ni guardan un registro histórico. El motivo de esto es

que conlleva complejidad adicional a la aplicación. Ahora, con la tecnología de agentes e Internet, es posible que un agente en la máquina del cliente se ocupe, el mismo, de llevar las estadísticas.
Hay factores importantes que el agente deberá poder aprender, por ejemplo un cliente rara ves compré un mismo tipo de zapatillas en el mismo mes; si esto ocurre, pasará después de un cierto tiempo. Relaciones de este tipo, suelen haber muchas, en cualquier dominio. Transcribimos aquí algunas de las más interesantes, y mostramos la relación estadística entre ellos.

En las tablas siguientes se detallan algunas de las relaciones entre atributos más importantes; y que habrá que tener en cuenta al momento de testear las implementaciones.

Marca con Categoría de Producto

En la *Tabla A.3* puede verse la relación existente entre algunas marcas y la categoría del producto, es decir hay determinadas marcas fuertemente relacionados con un tipo de producto, por ejemplo el caso de Gilbert y accesorios, donde el 100 % de los productos Gildbert son accesorios. Lo mismo pasa con las marcas más conocidas, como Niké y Adidas, estas se relacionan más con el calzado y la indumentaria que con otras categorías.

Marca/Categor	Accesorios	Calzado	Indumentaria	Entrenamiento	Porcentaje
Adidas	3,45	54,00	42,55	0,00	100,00
Asics	4,50	50,50	45,00	0,00	100,00
Easton	74,00	0,00	26,00	0,00	100,00
Franklin	97,00	0,00	3,00	0,00	100,00
Gilbert	100,00	0,00	0,00	0,00	100,00
Head	2,30	89,00	8,70	0,00	100,00
L.A Gear	0,00	79,40	20,60	0,00	100,00
Le Coq Sportif	2,43	13,40	84,17	0,00	100,00
Nike	4,68	55,32	40,00	0,00	100,00
Penalty	50,00	27,40	22,60	0,00	100,00
Reef	0,00	95,43	4,57	0,00	100,00
Signia	35,00	6,00	59,00	0,00	100,00
Spalding	4,00	0,00	0,00	96,00	100,00
Topper	1,03	84,43	14,54	0,00	100,00
Vairo	33,43	48,55	18,02	0,00	100,00

Tabla A.3 - Correlación Marca con Categoría de producto

Categoría Comprador con Deporte

En el siguiente gráfico se observa la distribución de las ventas, según el deporte con el que está relacionado el producto y la categoría del comprador. Vemos cómo hay categorías más relacionadas con el sexo y la edad que otros. Un Claro ejemplo es el de hombre con fútbol. Existen otras relaciones interesantes como adolescente con padle, y dama con fitness. Según puede observarse en el gráfico de la *Tabla A.4*

Precio / Exposición	Hombre	Dama	Niño	Adolescente	Porcentaje
Básquet	67,40	6,50	10,80	15,30	100,00
Béisbol	88,90	0,00	0,00	11,10	100,00
Fitness	35,90	43,30	5,40	15,40	100,00
Fútbol	40,70	6,50	27,20	25,60	100,00
Paddle	45,60	0,00	0,00	54,40	100,00
Rugby	54,53	0,00	17,20	28,27	100,00
Tenis	35,30	40,34	12,20	12,16	100,00
Natación	17,90	35,30	24,50	22,30	100,00

Tabla A.4 - Correlación Categoría Comprador con Deporte

Precio con Exposición

El precio se encuentra íntimamente ligado al tiempo de exposición. En la *tabla A.5* puede verse como a medida que transcurre el tiempo en vidriera del producto, éste disminuye su valor.

Precio / Exposición	Liquidación	Oferta	Normal	Sobreprecio	Porcentaje
Un Mes	0,80	3,40	67,80	28,00	100,00
Dos Meses	0,98	4,80	75,43	18,79	100,00
Tres Meses	1,20	8,90	78,80	11,10	100,00
Seis Meses	1,23	17,87	65,20	15,70	100,00
Mas de seis Meses	7,80	24,80	60,90	6,50	100,00

Tabla A.5 - Correlación Precio con Exposición

A.4. Cometarios Finales

A través de este breve análisis realizado, es posible comprender la complejidad del dominio, la cantidad de reglas, y relaciones en juego, entre todos los atributos que lo forman.
Muchas de las conclusiones, y análisis incluidos en este apéndice, fueron tenidas en cuenta a lo largo de la implementación del presente trabajo. Además este apéndice nos facilitó la selección de casos de prueba finales, así como también, el desarrollo de cada una de las técnicas; gracias a este análisis pudimos comprobar en todo momento, que técnicas estaban haciendo las cosas bien, y cuáles no; permitiendo, también, dar paso a la modificación de la técnica, y la búsqueda de otras aproximaciones, en los casos en los que estas no brindaban resultados adecuados.

Apéndice B

B. Un Framework Para Implementar Agentes de Interfaz en Entornos de Comercio Electrónico

Como dijimos a lo largo del presente trabajo, tenemos que implementar un agente de interfaz con distintas técnicas de aprendizaje. Esta tarea podría realizarse de la siguiente manera: implementando cada vez el agente para cada técnica a probar, y hacer la implementación dependiente de dominio. Pero si bien esto puede llegar a ser rápido de implementar, si en vez de tres técnicas queremos probar 100 técnicas, o variaciones de ellas, la tarea se vuelve más complicada. Si a esto lo sumamos que queremos realizar un almacenamiento estadístico de los tiempos de aprendizaje y consulta de cada técnica, independientemente de la técnica, y si le agregamos que deseamos probar el agente en otros entornos distintos de la tienda de comercio virtual en la cual lo estamos probando actualmente; nos damos cuenta la opción más lógica es la de realizar la implementación de un framework, cuyos objetivos sean los siguientes.

- Que permita parametrizar, la técnica de aprendizaje a utilizar.
- Que sea independiente del dominio de la aplicación, aunque siempre la aplicación será relativa a comercio electrónico.
- Que podamos realizar el testeo y el aprendizaje de varias técnicas a la vez, con la misma lección. Es decir que dada las lecciones recibidas por el usuario puedan, potencialmente, generarse varios perfiles de usuario, uno para cada técnica de aprendizaje.

- Que genere automáticamente las estadísticas para cada técnica de aprendizaje, y que a su vez, estas estadísticas puedan almacenarse en cualquier medio.
- El framework deberá permitir de manera sencilla ser extendido para que puedan almacenarse consultas y casos de test para ser ejecutados en conjunto. De esta manera se podrán testear las técnicas de aprendizaje en su conjunto; facilitando la contrastación de las nuevas técnicas implementadas, comparándolas, al ejecutar los mismos casos de test que a las técnicas ya implementadas.

B.1. Diseño e Implementación

El framework fue implementado íntegramente en java. En la siguiente figura se presenta un modelo abstracto del diseño del framework. El diagrama muestra únicamente, una aproximación abstracta de la implementación del framework, y tiene como único fin, dar una idea del funcionamiento de éste y cómo hace posible los objetivos planteados al principio.

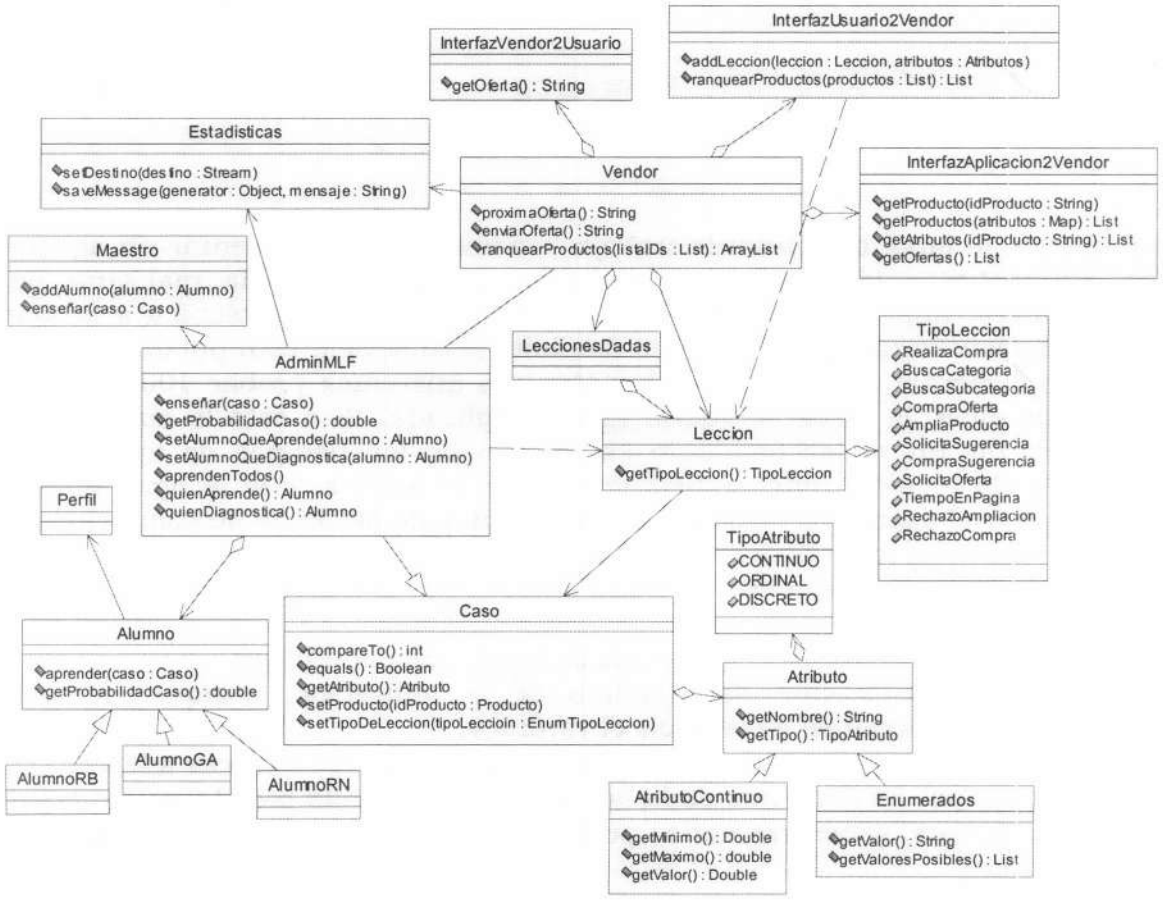

Figura B.1 - Arquitectura del Agente

A continuación describimos cada una de las clases y su función principal para entender cuál es su aporte y función.

B.2. Descripción de las clases

Vendor

Vendor es el agente propiamente dicho, él se encarga de manejar la lógica del agente. Vendor aprende al recibir una lección; se encarga de seleccionar ofertas según cuales sean las acciones del usuario. A su vez, Vendor brinda interfaces de comunicación entre él y el entorno que lo rodea, estas interfaces son, Vendor a Usuario (InterfaceVendor2Usuario), Usuario a Vendor (InterfaceUsuario2Vendor), y Aplicación a Vendor (InterfaceAplicación2Vendor); las que se describen más adelante.

AdminMLF

AdminMLF se encarga de mantener y controlar todas las acciones de los alumnos, él toma una lección del agente y se la pasa a los alumnos que estén aprendiendo en este momento.

Alumno

Un alumno es simplemente una interfaz, es decir un conjunto de métodos que tienen que implementar quienes deseen crear un nuevo alumno que aprenda mediante alguna técnica de aprendizaje. Basta con que implemente esos métodos, y luego podrá ser tratado por las demás clases de igual manera que los otros alumnos.

InterfaceVendor2Usuario

Esta interfaz es utilizada por *Vendor* para comunicarse con el módulo usuario, típicamente se encarga de decirle al usuario que *Vendor* tiene una nueva oferta para recomendar.

InterfaceUsuario2Vendor

El fin de esta interfaz es comunicar a *Vendor* ciertas acciones y pedidos del usuario.

InterfaceAplicacion2Vendor

Mediante ésta la aplicación tiene una forma de comunicarse con el agente. Es decir cualquier aplicación que implementé esta interfaz, puede utilizar el framework.

Estadísticas
Módulo encargado del almacenamiento de las acciones de los distintos objetos que conforman los módulos de aprendizaje, y el agente, propiamente dicho.

Maestro
Brinda una implementación abstracta para utilizar por cualquiera que quiera usar un nuevo administrador de alumnos (AdminMLF), de esta manera el corazón de la relación alumno – maestro queda implementada, independientemente de las reglas de mayor abstracción que quieran agregarse.

Atributo -TipoAtributo
Es un típico atributo posible del producto (precio, calidad, etc), estos tienen un tipo, que será, continuo, discreto, u ordinal; y queda especificado por la clase *TipoAtributo.*

AtributoContinuo – Enumerados

Ambos son atributos y tienen un nombre que los identifica la diferencia es que; un atributo continuo es sencillamente eso, un valor real referente a algún atributo continuo; mientras que un enumerado es un valor discreto con distinto valore posibles.

Lección -TipoLeccion

Una lección consta de un tipo de lección entre los tipos posibles, y un *Caso,* conteniendo la información necesaria para procesar la lección. Un Tipo de lección es una de las posibles acciones que realizó el usuario y que sirve como lección al agente (*Vendor).*

Caso
Un caso es ni más ni menos, que un conjunto de atributos. Sirve para indicar que atributos son visibles por el usuario en este momento; y sirve como interfaz entre los módulos de aprendizaje y la aplicación.

AlumnoGA
Es un alumno que aprende utilizando la técnica de algoritmos genéticos.

AlumnoRN
Es un alumno que aprende utilizando la técnica de redes neuronales.

AlumnoRB
Es un alumno que aprende utilizando la técnica de redes bayesianas.

B.3. Cometarios Finales

Creemos que este framework resulta muy poderoso si queremos seguir realizando contrastaciones de algoritmos en distintos ambientes. El desarrollo del presente framework es susceptible de varias mejoras. Creemos que éstas serán realizadas en futuros trabajos, cuándo necesitemos testear otros dominios de comercio electrónico.

Apéndice C

C. Referencias

[AGE/1995] http://agents.www.media.mit.edu/groups/agents/ Grupo de investigación de agentes con especial interés en los agentes interfaz.

[AMA/1999]D. Cordero, P. Roldán, S. Schiaffino, A. Amandi - "Intelligent Agents Generating Personal Newspapers" - Proceedings del ICEIS '99 - International Conference on Enterprise Information Systems - Setúbal, Portugal - Marzo 1999

[APA/1999] Jain A., Aparicio M. Singh M. "Agents for process coherence in Virtual Enterprises". Comunication of ACM, Mar 1999.

[BAY/1763] An essay towards solving a problem in the doctrine of changes. Philosofhical trnsactions of the royal Society of London,53:370-418.

[BEE/1994] Beerud Dilip Sheth, Phd: A Learning Approach to Personalizaed Information Filtering. MIT 1994 - Certified by Pattie Maes.

[BRA/1998] Jeffrey M. Bradshaw, "Software Agents", 1998.

[CHE/1995] David Chess, Colin Harrison, Aaron Kershenbaum, "Mobile Agents: Are They a Good Idea?", IBM Research DivisionT.J. Watson Research Center Yorktown Heights, New York, 1995

[DAM/1999]D'ambrosio, B:1999 Inference in Bayesian Networks. AI Magazine pp. 21-35

[DOU/1989]Dougherty, J., Kohavi, R., Sahami, M., Supervised and Unsupervised Discretizacion of Continuos Features - Computer Science Department, Standford University, 1989.

[DRA/1999] Tzvetan Drashansky et al. "Networked Agents for Scientific Computing". Comunication of ACM, Mar 1999.

[FA/1996] Fa-Chun Cheong "Internet Agents", New Readers, 1996.

[FRA/1996] S.Franklin, A. Graesser, "Is it an Agent, or just a program? A Taxonomy for Autonomous Agents", Proceeding of the Third International Workshop on Agent Theories, Architecture, and Languages, Springer-Verlag, 1996.

[FRI/1996] Kohavi, R, Friedman, J,Yun, Y. Lazy Decision Trees, to appear in AAAI-96

[GAN/2002] Gantovnik, V.B. and Gurdal, Z. and Watson, L.T. (2002) Genetic Algorithm with Memory for Optimal Design of Laminated Sandwich Composite Site Panels. Technical Report TR-02-07, Computer Science, Virginia Tech

[GAR/2000] T. Garcia. Aspectos Filosoficos del impacto de la auto-organización en la teoria de la evolucion. http://www.sc.ehu.es/sfwpbiog/Thesis-Tomas.html

[GLU/1999] Glushko R., Tenenbaum J. Meltzer B.. "An XML framework for agent-based E-commerce". Comunication of ACM, Mar 1999.

[GME/2001]Genetic Algorithm and Traveling Salesman Problem http://www.generation5.org/aisolutions/tspapp.shtml

[GOD/2000] Godoy, D., Amandi, A.: PersonalSearcher: An Intelligent Agent for Searching Web Pages. IBERAMIA-SBIA 2000 - Lecture Notes in Artificial Intelligence LNAI 1952 - Springer Verlag - pp 43 -52 - (2000)

[GOL/1987]D. E. Goldberg and J. J. Richardson. Genetic algorithms with sharing for multimodal function optimization. In ICGA87, 1987.

[GOL/1989 II]David E. Goldberg. Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley Publishing Company, Inc., 1989.

[GOL/1989.2]Kalyanmoy Deb and David E. Goldberg. An investigation of niche and speies formation in genetic function optimization. In Proc. of 3rd ICGA'89, 1989.

[GRO/1987] Grossberg, S. (1987). Competitive learning: from interactive activation to adaptative resonance. Cognitive Science, 11, 23-63.

[GRO/2000] Genetic Programming (JGProg) - http://jgprog.sourceforge.net/about_jgprog.html

[GUI/1995] Guilfoyle, "Vendors of Agent Technology", UNICOM Seminar on Intelligent Agents and their Business Applications, 1995.

[GUS/1999] Rune Gustavsson. "Agents with Power". Comunication of ACM, Mar 1999.

[HAD/199]Haddawy,p: 1999, An overview of some recent developments in Bayesian problem-Solving techniques AI Magazine pp 11-19

[HAY/1989]B. Hayes-Roth, M. Hewett, R. Washington, R. Hewett, and A. Seiver. Distributing intelligence within an individual. In L. Gasser and M. Huhns, editors, Distributed Artificial Intelligence Volume II. Pitman Publishing: London and Morgan Kaufmann: San Mateo, CA, 1989, pp. 385-412.

[HER/--] Björn Hermans, "Intelligent Software Agents on the Internet" (Tesis).

[HER/1990] John Hertz, Anders Krogh, Richrd G Palmer, "Introduction to the theory of neural computation" Addison-Wesley Publishing Company, Inc., 1991.

[HIL/1996] Hylton J., Manheimer K., Drake Jr. et al "Knowbot programming: System support for mobile agents". In Proceedings of the Fifth International Conference on Object Orientation in Operating Systems", Oct. 1996.

[HIN/1986] Hinton, G.E y Sejnowski, T.J. (1986) Learning and relearning in Boltzman machines. En Rumerlhart & McClelland (1986)

[HOL/1989] Holland, J.H., Holyoak, K.J., Nisbett, R. E., Thagard, P. R. Induction - Processes of Inference, Learning, and Discovery, Cambridge, MIT Press, 1989.

[HOP/1982] Hoplfield, J.L. (1982). Neural networks and physical systems with emergent collective computational abilities. Proceedings of the National Academy of Science USA, 79, 2554-2558

[HYA/1999] Hyacinth S. "Software Agents: An Overview".paper of Intelligent Systems Research AA&T, BT Laboratories - To appear in Knowledge Engineering Review, 1996

[IBM/1997] "Mobile Agent Facility Specification", General Magic, Inc. IBM Corporation, 1997

[JAV/2002] http://www.cs.cmu.edu/javabayes/home The javaBajes Proyect.

[JOO/2002] A Neural Net framework to create, train and test Neural Nets - http://joone.sourceforge.net/

[JOS/1999] Joshi A. and Sing M. "Multiagents system on the net". Comunication of ACM, Mar 1999.

[KIT/1997] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, and E. Osawa, "Robocup: The robot world cup initiative,"In Proceedings of the First International Conference on Autonomous Agents, (Marina Del Rey, California),pp. 340-347, February 1997.

[KOH/1984] Kohonen, T. (1984). Self-organization and associative memory, Series in Information Sciences, vol. 8. Berlin: Springer-Verlag

[KOH/1989] Kohavi, R.; Dougherty, J., Sahami, M., Supervised and Unsupervised Discretizacion of Continuos Features - Computer Science Department, Standford University, 1989.

[KOH/1995]Kohavi, R., Becker, B., Sommerfield, D. Improving Simple Bayes.Data Mining and Visualization Group Silicon Graphics Inc.

[KOH/1996]Kohavi, R, Wolpert, D. Bias Plus Variance Decomposition for Zero-One Loss Functions, 1996.

[KOH/1997]Kohavi, R., Becker, B., Sommerfield, D. Improving Simple Bayes.

[KOH/1999]Kohavi, R, Bauer, E. An Empirical Coparison of voting Classification Algorithms: Bagging, Boosting, And Variants, 1999

[KOH/2001] Kohavi Ron; Provost Foster. "Applications of data mining to electronic commerce" Kluwer Academic Publishers, 101 Philip Drive, Assinippi Park, Norwell, Massachusetts 2001

[LAW/2001] Laerence R, Almasi G, Kotlyar V, Viveros M, Duri S. "Personalization of Supermarket product Recommendatios" paper of "Applications of data mining to electronic commerce" Kluwer Academic Publishers, 101 Philip Drive, Assinippi Park, Norwell, Massachusetts 2001

[LEE/2001] Visualization and Analyss of Clickstream Data of Online Stores for Understanding Web Merchandising. Juhnyoung Lee, Mark Podlaseck, Edith Schonberg, Robert Hoch paper of "Applications of data mining to electronic commerce" Kluwer Academic Publishers, 101 Philip Drive, Assinippi Park, Norwell, Massachusetts 2001

[LIE/1995] Lieberman, H "Autonomous Interface Agents", Massachusetts Institute of Technology Cambridge, Mass 02139 USA.

[MAE/1994] Maes P "Agents that reduce work and information overload". Communications of the acm, vol. 37 Nro. 7, 31-40.

[MAE/1999] P. Maes, R. Guttman., Moukas A. "Agents that buy and sell". Comunication of ACM, Mar 1999.

[MAR/1997] Martin, D., Oohama, H., Moran , D., Chever, A.: Information Brokering in an agent Architecture - in Proceedings of the Practical Application of Intelligent Agents and Multiagent Technology 1997 - PAAM 97 (1997) 467 – 486

[MIT/1997] Mitchell, T. M. "Machine learning," Boston, McGraw-Hill, 1997.

[MOS/1999] Moses Ma. "Agents in E-commerce". Comunication of ACM, Mar 1999.

[NAR/1999] Nardi B., Miller J. Wright D. "Interface Agents: Apple Data Detector". Comunication of ACM, Mar 1999.

[NWA/1996] Nwana Hyacinth S., Software Agents: An overview. Intelligent Systems Research, AA&T, BT Laboratories, 1996.

[OAT/1997] Oates., Jensen, : The Effects of Training Set Size on Decision Tree Complexity, 1997.

[OSM/1995] Osman, I. An Introduction to Meta-Heuristics. Operational Research Tutorial Papers Series, Annal Conference OR37 - Canterbury, 1995.

[PAR/1987] H. Van Dyke Parunak. Manufacturing experience with the contract net. In M. Huhns, editor, Distributed Artificial Intelligence. Pitman Publishing: London and Morgan Kaufmann: San Mateo, CA, 1987, pp.285-310.

[PAZ/2000] Pazzani M., Billsus D. "Evaluating Adaptative Web Site".University of California Irvine. - URL: http://www.ics.uci.edu/~pazzani/Publications

[PET/1996] C. Petrie, "Agent-Based Engineering, the Web, and Intelligence", 1996.

[PET/1999] C. Petrie, S. Goldman, "Agent-Based Project Management", 1999.

[QUI/1991] Quinlan,(1991). Connectionism and Psychology. Harvester Wheateaf. N.Y.

[ROS/1962] Rosemblatt, R. (1962) Principles of neurodynamicas. New York: Spartan Books.

[ROT/1989] B. Hayes-Roth, M. Hewett, R. Washington, R. Hewett, and A. Seiver. Distributing intelligence within an individual. In L. Gasser and M. Huhns, editors, Distributed Artificial Intelligence Volume II. Pitman Publishing: London and Morgan Kaufmann: San Mateo, CA, 1989, pp. 385-412.

[RUM/1986]Rumelhart, D.E., McClelland, J.L. & Group, PR (1986) Parallel Distributed Processing. Explorations in the Microstructure of Cognition. Cambridge, MA: MIT Press

[RUS/1995] Russell, S., Norvig, R.: "Artificial Intelligence - A modern approach", Upper Saddle River, New Jersey, Prentice-Hall, 1995.

[SCH/2001] Silvia N. Schiaffino, Analía Amandi. The QueryGuesser Agent Isistan Research Institute, FCE -UNICEN Buenos Aires. Argentina.

[SLO/1999] Sloman A and Logan B. "Building Cognitively Rich Agents using Sim_agent toolkit". Comunication of ACM, Mar 1999.

[SOR/1995] Sorensen, H. Mc Elligott, M. PSUN: A Profiling System for Usenet News, CIKM'95 Intelligent Information Agents Workshop, Baltimore, December 1995.

[SUS/1997] Sushil J. Louis and Gan Li. Combining Robot Control Strategies using Genetic Algorithms with Memory. Department of Computer Science - University of Nevada -Reno - 89557.

[SYC/1994] Sycara, K. and Zeng, D. "Towards an Intelligent Electronic Secretary". In Proceedings of the CIKM-94 (International Conference on Information and Knowledge Management). Workshop entitled, "Intelligent Information Agents." Gaithersburg, MD, 1994.

[SYC/1996] Hattori F., Ohguro T., Yokoo M., Matsubara S and Yoshida S. "Socialware Multiagents systems for supporting network comunities". Comunication of ACM, Mar 1999.

[SYC/1996.2] Sycara, K. and Zeng, D. "Multi-Agent Integration of Information Gathering and Decision Support". In the Proceedings of the European Conference on Artificial Intelligence. 1996.

[SYC/1996.3]Sycara, K., Decker, K. and Williamson, M. Modeling Information Agents: Advertisements, Organizational Roles, and Dynamic Behavior. Working Notes of the AAAI-96 Workshop, "Agent Modeling." Aug., 1996.

[SYC/1997.2] Sycara, K., Decker, K., Pannu, A. and Williamson, M. "Designing Behaviors for Information Agents". In Proceedings of the First International Conference on Autonomous Agents (AGENTS-97), Feb., 1997.

[SYC/1997] Jennings, Sycara, Wooldridge, "A Roadmap of Agent Research and Development".

[SYC/1997.3] Sycara, K., Decker, K., and Williamson, M. "Middle-Agents for the Internet". Proceedings of IJCAI-97, Jan., 1997.

[TSU/1998] Todd Sundsted, "An introduction to agents", June 1998.

[VAN/1987]H. Van Dyke Parunak. Manufacturing experience with the contract net. In M. Huhns, editor, Distributed Artificial Intelligence. Pitman Publishing: London and Morgan Kaufmann: San Mateo, CA, 1987, pp.285-310.

[WAS/1989] Wassermann, P.D. (1989). Neural computing: Theory and Practice. VNR. New York

[WON/1999] Wong D., Paciorek N. And Moore D.. "Java-based Mabile Agents". Comunication of ACM, Mar 1999.
Software Agents: A review, May 1997, Trinity College Dublin.