B E N E M É R I T A

UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

"TOPOLOGÍA DE UNA RED NEURONAL ARTIFICIAL MEDIANTE GANANCIA DE INFORMACIÓN"

TESIS

QUE PARA OBTENER EL TÍTULO DE:

LICENCIADA EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA:

DALIA RODRÍGUEZ SALAS

ASESORES:

DR. JOSÉ ARTURO OLVERA LÓPEZ

DRA. MARÍA DEL PILAR GÓMEZ GIL
INSTITUTO NACIONAL DE ASTROFÍSICA, ÓPTICA Y ELECTRÓNICA

PUEBLA, PUEBLA.

JULIO 2012

AGRADECIMIENTOS

- *A mi familia nuclear; mis padres por su apoyo incondicional, su confianza, sus enseñanzas y su comprensión; a mis hermanos mayores y mi sobrina Greta; a mi hermana Violeta que siempre ha caminado a mi lado. Ustedes son y serán un motivo para vencer adversidades.*

- *A mis familiares, los que están y aquellos que se han ido son un aliciente en mi vida.*

- *A Jorge, por estar a mi lado y ser quien es.*

- *A mis amigos por coincidir en mi camino y hacerlo más ameno.*

- *A mis profesores por contribuir en mi formación profesional. En especial al Lic. Jehú Martínez Espinosa, al Dr. Carlos Guillen Galván, al M.C. Ismael González Tzontecomani, a la Dra. Rosa García Tamayo, y a la M.C. Hilda Castillo Zacatelco.*

- *Al Dr. David Pinto Avendaño por su apoyo durante mi servicio social.*

- *Al Dr. Abraham Sánchez López por su apoyo durante la realización de esta tesis.*

- *A mi profesor y asesor de tesis, el Dr. José Arturo Olvera López a quien admiro por su gran vocación como docente y conocimientos en el área, por guiarme de principio a fin en el desarrollo de esta tesis.*

- *A mi asesora de tesis, la Dra. María del Pilar Gómez Gil quién confió en mí, me apoyó, enseñó y guió durante el desarrollo de esta tesis, tanto en el aspecto profesional como el personal.*

A todos y cada uno, les agradezco de la manera más sincera, haber influido positivamente en mi vida.

RESUMEN

En las redes neuronales artificiales MLP *(Multi-Layer Perceptron)*, por lo regular, las entradas para cada una de las neuronas de la primera capa oculta, son los atributos que definen los objetos o ejemplos de entrenamiento. Las salidas de estas neuronas serán la entrada para cada neurona de la siguiente capa, generándose así la alimentación de las neuronas hacia delante, hasta llegar a la capa de salida.

En este trabajo de tesis se propone un modelo de conexión entre neuronas de una red MLP, en el cual existe un menor número de conexiones respecto al modelo estándar, lo que permite reducir costos computacionales. La manera de conectar neuronas en esta propuesta, está basada en una métrica utilizada en la construcción de árboles de decisión llamada *Ganancia de Información*. Las redes construidas bajo este modelo las hemos llamado *IG nets* (*Information Gain nets*).

ÍNDICE GENERAL

CAPÍTULO1. INTRODUCCIÓN 1

CAPÍTULO 2. MARCO TEÓRICO Y TRABAJO RELACIONADO 5

2.1. ALGORITMO BACKPROPAGATION 6

2.2. ÁRBOLES DE DECISIÓN 9

2.2.1. GANANCIA DE INFORMACIÓN 10

2.2.2. ALGORITMO ID3. 12

2.3. REDES PARCIALMENTE CONECTADAS. 13

2.3.1. ENTROPY NETS 14

CAPÍTULO 3. TOPOLOGÍA DE UNA RED MLP BASADA EN GANANCIA DE INFORMACIÓN 16

3.1. DESCRIPCIÓN DEL MODELO 17

3.2. IMPLEMENTACIÓN 20

3.2.1. CONSTRUCCIÓN DEL ÁRBOL DE DECISIÓN 21

3.2.2. EXTRACCIÓN DE INFORMACIÓN 22

3.2.3. CONSTRUCCIÓN DE LA IG NET 23

CAPÍTULO 4. PRUEBAS Y RESULTADOS 25

4.1. DESCRIPCIÓN DEL EXPERIMENTO 26

4.1.1. PRIMERA FASE 27

4.1.2. SEGUNDA FASE 29

4.2. ANÁLISIS DE RESULTADOS 32

CAPÍTULO 5. CONCLUSIONES Y TRABAJO FUTURO 33

5.1. TRABAJO FUTURO 34

APÉNDICE A. LA CLASE "*CLASSIFIERTREE*" 35

APÉNDICE B. DESCRIPCIÓN DE ESCENARIOS 37

A.1. REQUERIMIENTOS DE USUARIO 38

A.2. REQUERIMIENTOS ESPECÍFICOS 38

A.3. DIAGRAMA DE CASOS DE USO 38

A.4. DESCRIPCIÓN DE ESCENARIOS DE CASOS DE USO 39

APÉNDICE C. VERIFICACIÓN DE LA IMPLEMENTACIÓN DE MODELOS 44

B.1. MODELO IG NET 45

B.1. MODELO ENTROPY NET 46

APÉNDICE D. MANUAL DE USUARIO 48

BIBLIOGRAFÍA. 51

ÍNDICE DE FIGURAS

Figura 1. Modelo de una neurona artificial [2000, Amit Konar]. 2
Figura 2. Red MLP con dos capas ocultas y topología totalmente conectada. 6
Figura 3. Ejemplo de un árbol de decisión. 10
Figura 4. Gráfica de la entropía para dos clases (+ y -). 11
Figura 5. Pseudocódigo para el algoritmo ID3. 13
Figura 6. a) Árbol de decisión correspondiente al problema XOR. **b)** Entropy Net asociada al problema XOR. 15
Figura 7. a) Árbol de decisión mostrado en la figura 3 para un problema de 3 clases y tres atributos. **b)** IG net asociada a nuestra propuesta. **c)** Entropy net asociada a a). Los colores en los nodos del árbol y la red representan las clases. 20
Figura 8. Diagrama de clase con las relaciones entre las clases utilizadas de la librería Weka para la implementación del modelo *IG net*. 22
Figura 9. a) Árbol de decisión mostrado en la figura 3. **b)** Estructura de los datos extraidos del árbol. Las Xi son los atributos y Ci corresponde a las clases. 23
Figura 10. a) Obtención de capas ocultas utilizando información del árbol. **b)** Regla R5 aplicada a a). 24
Figura 11. Interfaz gráfica del sistema que implementa el modelo de las *IG nets*. 24
Figura 12. Promedio del porcentaje de clasificación al realizar validación cruzada con 10 *folds*. 29
Figura 13. Gráfica de barras para el tiempo de entrenamiento de cada modelo de red para cada conjunto de datos. 30
Figura 14. Gráfica de barras para el número de conexiones de cada modelo de red para cada conjunto de datos. 30
Figura 15. Número de neuronas ocultas en las redes construidas con modelo *IG net* y *Entropy net* para los conjuntos de datos de la tabla 1. 31
Figura 16. Principales atributos y operaciones de la clase *ClassifierTree.java.* El símbolo '*' indica atributos y operaciones agregadas para la implementación del modelo *IG net*. 36
Figura 17. Diagrama de casos de uso del sistema PCN-Tree. 38
Figura 18. Interfaz gráfica del sistema PCN-Tree, indicando objetos que permiten ejecutar los requerimientos. 43
Figura 19. a) Descripción del árbol de decisión construido por la librería Weka para el modelo IG Net. **b)** Representación gráfica de a). 45
Figura 20. a) Descripción de la red construida por el sistema PCN-Tree con modelo IG Net. **b)** Representación gráfica para a). 46
Figura 21. a) Descripción del árbol de decisión construido por la librería Weka para el modelo Entropy Net. **b)** Representación gráfica de a). 46
Figura 22. a) Descripción de la red construida por el sistema PCN-Tree con modelo Entropy Net. **b)** Representación gráfica para a). 47

Figura 23. a) Interfaz gráfica del sistema PCN-Tree. **b)** Ventana de configuración de parámetros de entrenamiento.....49

ÍNDICE DE TABLAS

Tabla 1. Descripción de los conjuntos de datos utilizados para los experimentos.....27
Tabla 2. Porcentaje de clasificación al evaluar cuatro modelos de redes MLP utilizando validación cruzada con 10 *folds* para el conjunto de datos Iris.....28
Tabla 3. Medias y desviaciones estándar de los porcentajes de clasificación obtenidos de realizar validación cruzada con 10 *folds* para evaluar cuatro modelos de redes MLP con los conjuntos de datos descritos en la tabla 1.....28
Tabla 4. Tiempo de entrenamiento (t) medido en segundos y número de conexiones (c) para cada uno de los cuatro modelos de red MLP construidos con los conjuntos de datos de la tabla 1.....29
Tabla 5. Comparación del número de conexiones y tiempo de entrenamiento utilizando la prueba WRS. El símbolo '*' representa qué modelo fue significativamente mejor.....31

CAPÍTULO 1

INTRODUCCIÓN

Los primeros modelos del funcionamiento de una neurona biológica fueron realizados en [1943, McCulloch et al.], tras varios años de investigación donde hubo resultados desalentadores acerca de este tipo de modelos. Su uso fue más notorio con la publicación realizada en [1986, Rumelhart et al.] donde se propone un algoritmo de aprendizaje para una red neuronal con varios niveles utilizando retro-propagación del error. A partir de este trabajo se han propuesto distintos modelos de redes neuronales artificiales [2000, Konar].

Las redes neuronales artificiales están inspiradas en las redes neuronales biológicas. Tanto en las redes biológicas como en las artificiales, las neuronas se encuentran interconectadas a neuronas vecinas. En las redes neuronales artificiales, una neurona también es llamada *perceptrón* (ver figura 1), la manera en que se encuentran conectadas, define la topología de la red.

Existen distintos modelos de redes neuronales artificiales, los cuales pueden ser cíclicos o acíclicos. Las redes con modelo cíclico son llamadas *redes recurrentes* y tienen como principal característica que las salidas de cada neurona pueden ser sus propias entradas, a diferencia de las redes con modelo acíclico, llamadas *redes feed-forward (con alimentación hacia delante)*, donde las señales de salida alimentan a otras neuronas, y no a ellas mismas.

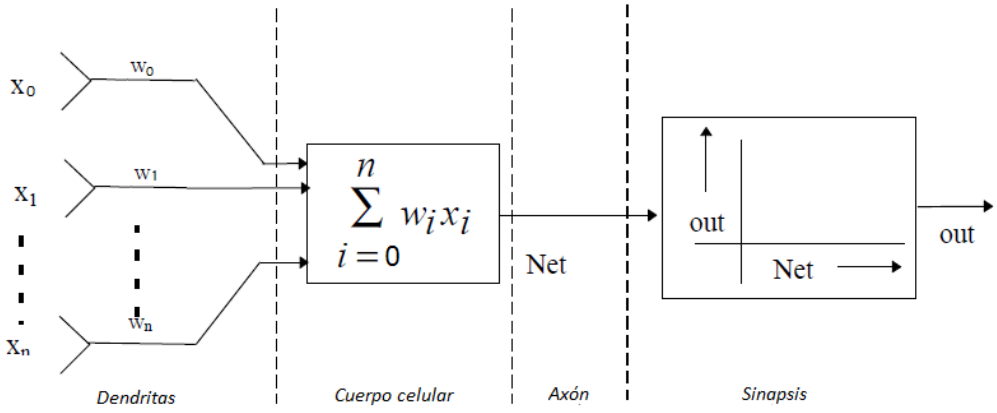

Figura 1. Modelo de una neurona artificial [2000, Amit Konar].

La redes *feed-forward* normalmente están formadas por capas de neuronas, donde las salidas de las neuronas pertenecientes a una capa, serán las entradas de las neuronas de la capa siguiente, hasta llegar a la capa de salida. Cuando este tipo de redes tienen dos o más capas, estas redes son llamadas *Perceptrón Multicapa* o *"Multi Layer Perceptron"* (MLP). Este tipo de redes pueden ser entrenadas para aprender a categorizar objetos en clases identificadas, dándole a conocer ejemplos del tipo de objetos que debe categorizar, recibiendo en las entradas de la red las carac-

terísticas o atributos, que describen a los objetos y conociendo la clase a la que pertenecen con la finalidad de *supervisar* el aprendizaje obtenido por la red, teniendo como resultado un clasificador. A esta habilidad de las redes se le llama Mapeo Entrada-Salida o IOM de las siglas en inglés *"Input–Output Mapping".* Los objetos con los que se entrena la red, forman un conjunto de objetos conocido como *conjunto entrenamiento* [2009, Haykin]. Una de las principales características de estas redes es que una vez entrenadas para categorizar objetos, tienen el *conocimiento* para poder generalizar y clasificar objetos distintos con los que fueron entrenadas, siempre y cuando estén descritos por las mismas características de los objetos en el conjunto entrenamiento.

El uso de redes MLP para resolver problemas complicados IOM es común debido a las ventajas que de ellas se tienen, sin embargo, el costo computacional que requieren durante el proceso de aprendizaje es muy elevado debido en gran medida a su topología; reducir el costo computacional ha sido tema de interés para investigadores y se han desarrollado diversas propuestas. A raíz del intento de reducir costos computacionales en el proceso de aprendizaje, surgieron las redes MLP parcialmente conectadas, las cuales en su topología, tienen un número menor de conexiones entre neuronas respecto a la topología estándar.

En este trabajo de tesis se definió un proceso de construcción de la topología de una red neuronal parcialmente conectada mediante una métrica utilizada en la construcción de árboles de decisión llamada *ganancia de información* (basado en la teoría de Shannon y Weaver, 1949). Este proceso supone que las MLP son utilizadas para clasificación o problemas del tipo IOM.

Durante el proceso de aprendizaje se determina la relevancia de un atributo en una red MLP, lo que puede ser muy tardado, debido a que cada atributo es considerado varias veces en cada neurona. Esto ocurre porque las neuronas que procesan directamente los atributos de cada objeto envían estas señales a otras neuronas dónde el valor del atributo afecta indirectamente. Esto es diferente a la manera en que un árbol de decisión determina la relevancia de cada atributo, donde cada atributo es analizado una sola vez, proceso explicado en el capítulo 3.

El trabajo presentado aquí está basado en la idea de que, si la construcción de un árbol de decisión permite conocer la capacidad discriminante de un atributo para determinar la clase a la que pertenece un objeto, se puede considerar como herramienta para establecer las conexiones en una red neuronal artificial, obteniendo la ventaja de la propiedad de generalización que tiene una red.

La topología propuesta bajo este enfoque muestra resultados favorables al reducir el número de conexiones de manera significativa respecto a la topología estándar de una red MLP.

Este documento está organizado de la siguiente manera: en el capítulo 2 se plantea el marco teórico en el que se basa la topología propuesta, además de trabajos que ya se han realizado al respecto. En el capítulo 3 se define el modelo de la topología propuesta en esta tesis, presentando en el capítulo 4 los resultados al evaluar la topología en problemas IOM y finalizando en el capítulo 5 con las conclusiones y trabajo futuro que se propone, relacionado con este trabajo de tesis.

CAPÍTULO 2

MARCO TEÓRICO Y TRABAJO RELACIONADO

En este capítulo se plantean los conocimientos teóricos para el desarrollo de la topología de una red MLP mediante ganancia de información, además se presenta una breve introducción a los distintos enfoques que se han propuesto para eliminar conexiones innecesarias en la topología estándar. Se incluye un apartado para las *Etropy Nets* [1990, Sethi], una clase de redes construidas con un enfoque similar al planteado en esta tesis.

2.1. ALGORITMO BACKPROPAGATION

El algoritmo de retropropagación o *backpropagation* es utilizado para que una red MLP aprenda de manera supervisada, esto es, para cada objeto en el conjunto de entrenamiento se conoce la clase a que éste pertenece. Este algoritmo fue publicado inicialmente por [1974, Werbos] e independientemente desarrollado por [1986, Rumelthart et al.]. Este tipo de redes se utiliza para modelar problemas no lineales multiclase; la estructura común de una red MLP entrenada con retropropagación consiste en una capa de entrada, al menos una capa oculta y una capa de salida (ver figura 2).

Es importante resaltar que en la capa de entrada no hay neuronas procesadoras, solo se representan para indicar puntos de entrada. Cuando se usa una red MLP como clasificador, cada uno de estos puntos de entrada corresponde a un punto de conexión con cada componente del vector de características que describen al objeto. En las capas ocultas y en la capa de salida, cada nodo representa una neurona; el número de clases posibles en el conjunto entrenamiento será el número de neuronas que habrá en la capa de salida. En una topología totalmente conectada, cada nodo perteneciente a una capa, tendrá una conexión con cada neurona de la siguiente capa.

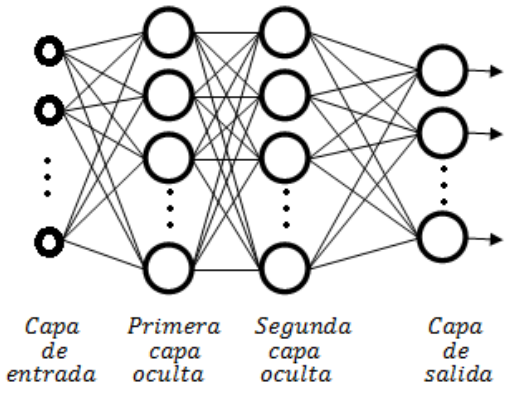

Figura 2. Red MLP con dos capas ocultas y topología totalmente conectada.

En el proceso de aprendizaje de retropropagación existen dos subprocesos. El primero es aquel en el cual se alimentan las neuronas hacia delante, empezando en la primera capa oculta,

donde cada neurona recibe los atributos de un objeto y calcula el valor de la señal de salida, llamada señal función, para ser enviada a cada neurona de la siguiente capa continuando este proceso capa por capa hasta llegar a la capa de salida. El segundo proceso empieza en la capa de salida, donde cada neurona calcula el error de acuerdo a la diferencia de la salida real con la deseada, generando una señal llamada señal de error, que es retropropagada a las neuronas de la capa anterior al mismo tiempo de corregir los pesos sinápticos [2009, Haykin].

Como se dijo anteriormente, en la capa de salida cada neurona está asociada con una clase; al finalizar el proceso de aprendizaje se espera que los pesos involucrados directa o indirectamente en el valor de salida de estas neuronas, generen los hiperplanos de separación de las posibles clases. Enseguida se explica el proceso de aprendizaje, basándose en la siguiente notación [2005, Pajares et al.]:

$w_{ji}^{k} =$ Peso sináptico que ponderará la salida de la $i-ésima$ neurona de la capa k en la $j-ésima$ neurona de la capa $k+1$.

Nota: Se asume que la capa de entrada es la capa 0.

$F =$ Función de activación, continua y diferenciable.

$$Net_j^k = \begin{cases} \sum_{i=0}^{N_{k-1}} w_{ji}^{k-1} O_i^{k-1} \text{ , si } k > 1 \\ \sum_{i=0}^{n} w_{ji}^{k-1} x_i \quad \text{ , si } k = 1 \end{cases} \tag{1}$$

donde O_i^{k-1} es la salida de la $i-ésima$ neurona de la capa $k-1$ y N_{k-1} es el número de neuronas en la capa $k-1$.

$O_j^k =$ $F(NET_j^k)$ Salida de la $j-ésima$ neurona de la capa k.

$\Delta w_{ji}^{k} =$ Incremento en el peso w_{ji}^{k} provocado por el $p-ésimo$ patrón.

$T_p =$ Vector de salida deseada del $p-ésimo$ patrón.

$\delta_j^k =$ Señal de error de la $j-ésima$ neurona en la $k-ésima$ capa.

$\eta =$ Tasa de aprendizaje.

Los pasos del algoritmo se describen a continuación [2005, Pajares et al.], [2011, Gómez-Gil]:

El proceso de aprendizaje comienza inicializando con números aleatorios los pesos sinápticos de cada neurona en la red.

Para cada vector de entrada X, perteneciente al conjunto entrenamiento, se define el correspondiente vector de salida deseada T_p. Cada entrada del vector T_p corresponde a la salida deseada de las neuronas en la capa de salida Q, cuando la red MLP se está usando como clasificador; si la clase del patrón definido en el vector de entrada es de clase C_k , entonces el elemento t_k del vector T_p será 1, en caso contrario será 0. Cuando la red MLP se usa como clasificador, la función de activación de las neuronas en la capa de salida es una sigmoide.

Se calculan las salidas O_j^k correspondientes a la primera capa oculta. Estas salidas alimentan a las neuronas de la siguiente capa hasta llegar a la capa de salida. Una vez obtenidas las salidas, se calcula el error E_p generado por el $p - ésimo$ patrón de la siguiente manera:

$$E_p = \frac{1}{2} \sum_{j=0}^{N_Q - 1} (t_j - O_j^Q) \tag{2}$$

Si la salida deseada fuera igual que la real, quiere decir que $E_p = 0$, por lo tanto no hay necesidad de ajustar los pesos; en caso contrario deben actualizarse, calculando cada incremento Δw_{ji}^k mediante la expresión:

$$\Delta w_{ji}^k = \eta \delta_j^k O_i^{k-1} \tag{3}$$

El proceso de retropropagación empieza calculando la señal de error δ_j^k de cada neurona en la capa de salida:

$$\delta_j^Q = \left(T_p - O_j^Q\right) * F'\left(NET_j^Q\right) \tag{4}$$

La señal de error de una neurona en la capa oculta, debe calcularse dependiendo de las señales de error de las neuronas que alimentó:

$$\delta_j^k = F'\left(NET_j^k\right) * (\sum_{m=1}^{N_{k+1}} \delta_m^{k+1} w_{mj}^{k+1}) \tag{5}$$

Conociendo el valor de las señales de error se puede calcular el valor de los Δw_{ji}^k utilizando la expresión (3) y entonces actualizar los pesos:

$$w_{ji}^{k} = w_{ji}^{k} + \Delta w_{ji}^{k} \tag{6}$$

El error total E_T es la suma de los errores E_p generados por cada patrón en el conjunto entrenamiento.

Se puede notar que el número de conexiones entre neuronas puede ser muy grande, debido a que cada neurona de la $k - ésima$ capa es conectada a cada neurona de la capa $k + 1$. Por otra parte, ajustar los pesos puede llegar a ser muy tardado, dependiendo de factores como lo son el número de objetos en el conjunto entrenamiento, la cantidad de características de cada objeto y el número de capas ocultas.

No hay una topología fija en este tipo de redes, el número de neuronas en las capas ocultas puede ser variado e incluso algunas conexiones entre neuronas pueden ser omitidas. Esto podría hacerse si se tiene conocimiento sobre el problema que se está modelando con la red, ya que una de las funciones de las neuronas ocultas es extraer progresivamente los atributos más significativos de los objetos pertenecientes al conjunto entrenamiento. Una tarea similar a esta tarea de escoger conexiones es realizada en la construcción de un árbol de decisión, donde se utilizan medidas de impureza y homogeneidad, para decidir qué características serán los nodos en cada nivel del árbol.

2.2. ÁRBOLES DE DECISIÓN

Un árbol de decisión, al igual que una red neuronal cuando se utiliza como clasificador, permite determinar a qué clase pertenece un objeto. Los árboles de decisión, para determinar la clase a la que pertenece un objeto, deben aprender reglas con la siguiente forma [2005, Pajares et al.], [2006, Sierra]:

$$\text{Si } (x_1 < u_1) \wedge (x_2 \geq u_2) \wedge \ldots \wedge (x_n \geq u_m) \text{ entonces } (C_k) \tag{7}$$

Donde

$x_i =$ $i - ésimo$ atributo.

$u_j =$ $j - ésimo$ umbral.

$C_k =$ $k - ésima$ clase.

Para encontrar dichas reglas, el árbol debe ser construido a través de los objetos en el conjunto entrenamiento, siguiendo un proceso recursivo que comienza en la raíz del árbol y termina en sus hojas. Los nodos que no son hojas representan atributos de los objetos, las ramas representan los valores de dichos atributos y las hojas son los valores de la clase seleccionada (ver figura 3).

La métrica para decidir qué atributo respecta a cada nodo del árbol, es llamada ganancia de información, la cual determina la homogeneidad de cada atributo, respecto a las clases. Un atributo es seleccionado, si tiene una mayor capacidad de discriminar la clase a la que pertenecen los objetos en el conjunto entrenamiento.

Es importante observar que el proceso de aprendizaje en un árbol de decisión se realiza de manera supervisada, es decir, para determinar las reglas de clasificación, se debe conocer la clase a la que pertenece cada objeto del conjunto entrenamiento.

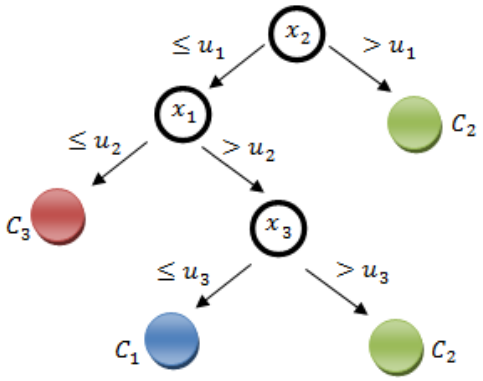

Figura 3. Ejemplo de un árbol de decisión.

2.2.1. GANANCIA DE INFORMACIÓN

La ganancia de información se auxilia de otra métrica que permite determinar el desorden que existe en un conjunto de datos, llamada entropía. La entropía está definida en la siguiente expresión [1997, Mitchell]:

$$\text{Entropy}(T) = \sum_{i=1}^{\check{N}_c} -p_i \log_2 p_i \tag{8}$$

Donde

$\mathrm{T} =$ Conjunto de entrenamiento.

$\check{\mathrm{N}}_{\mathrm{c}} =$ Número de clases en T.

$\mathrm{p_i} =$ Proporción de objetos en T pertenecientes a la i-ésima clase.

Y por definición $0 * \log_2 0 = 0$.

De la gráfica de la entropía (ver figura 4) se puede observar que, cuando los datos son homogéneos, es decir, todos pertenecen a la misma clase, la entropía es 0. Esto significa que si la entropía es una métrica para determinar la cantidad de desorden en un conjunto de datos entonces podemos decir que a mayor homogeneidad en los datos, menor es la entropía.

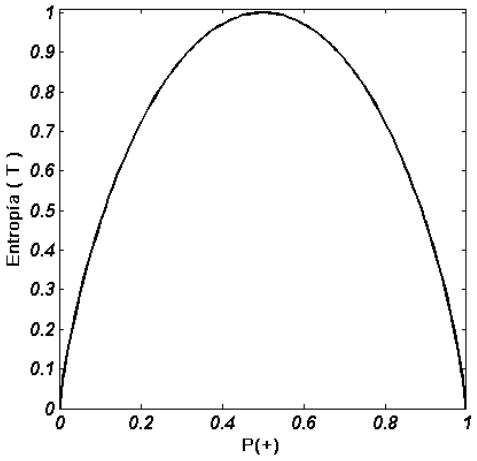

Figura 4. Gráfica de la entropía para dos clases (+ y -).

La ganancia de información de un atributo A respecto a un conjunto entrenamiento T se define mediante la siguiente expresión:

$$Gain(T, A) = Entropy(T) - \sum_{v \in Values(A)} \frac{|T_v|}{|T|} Entropy(T_v) \tag{9}$$

Donde

$|T_v| =$ Número de objetos con valor v en el atributo A.

$|T| =$ Número de objetos en el conjunto entrenamiento.

$Values(A) =$ Son los posibles valores de v para el atributo A.

$Entropy(T_v) =$ Entropía de aquellos objetos en T con valor v en el atributo A.

Los posibles valores de un atributo deben ser finitos para que la expresión (9) tenga sentido. En el caso de que los posibles volares de un atributo se encuentren en un intervalo con valores infinitos, como sería el caso de valores reales o enteros, se debe recurrir a un método de discretización tales como *Entropy Minimization Heuristic* [1993, Fayyad et al.] y 1RD [1993, Holte].

Las reglas de la forma (7) obtenidas al finalizar la construcción del árbol, describen a todos los objetos del conjunto entrenamiento. Cada objeto cumple con una única regla, la cual determina la clase a la que éste pertenece.

2.2.2. ALGORITMO ID3.

Para decidir qué atributo será la raíz del árbol, se calcula la ganancia de información que proporciona cada atributo de los objetos en el conjunto entrenamiento con la expresión (9). Posteriormente, se construye un nodo correspondiente al atributo con mayor ganancia de información, tal atributo se elimina de la lista de atributos. El nodo tendrá la misma cantidad de ramas como posibles valores v_i existen para dicho atributo.

Cada nodo se asocia a un conjunto de objetos de acuerdo al valor de atributo correspondiente a una rama, con los cuales se decidirá si al final de la rama habrá un nodo hoja. Se asigna un nodo como hoja en caso de cumplir con alguna de las siguientes condiciones:

- Todos los objetos pertenecen a la misma clase. En este caso la hoja corresponde a la clase a la que pertenecen los objetos.
- Si la lista parcial de atributos considerados durante la construcción del árbol está vacía. En este caso la hoja tendrá la clase a la que pertenezcan la mayoría de los objetos.

En caso de que el nodo no cumpla ninguna de estas dos condiciones, se repite el proceso recursivo de construcción del árbol. En la figura 5 se describe el algoritmo ID3 completo para la construcción de un árbol de decisión para el caso de dos clases (+ y -) [1997, Mitchell].

Existen varios algoritmos para la construcción de árboles de decisión, por ejemplo, C4.5 [1993, Quinlan], CART [1984,Breiman et al.], SPRINT [1996, Shafer et al.], SLIQ [1996, Mehta et al.] [2011, Kotsiantis]. El proceso para la construcción de árboles de decisión explicado en este documento está basado en el algoritmo ID3 [1987, Quinlan], el cual maneja exclusivamente atributos

con valores discretos. C4.5 es una variante de ID3 que permite la construcción de árboles a partir de atributos continuos y discretos.

Algoritmo ID3 ($Ejemplos$, $Clases$, $Atributos$)

Ejemplos **:** Objetos pertenecientes al conjunto de entrenamiento.
Clases **:** Valores posibles para la clase a seleccionar por el árbol {+,-}.
Atributos **:** Lista de atributos a comprobar por el árbol.

(1) Crear un nodo $raíz$ para el árbol.
(2) **Si** todos los ejemplos son positivos **Return** ($raíz$,+)
(3) **Si** todos los ejemplos son negativos **Return** ($raíz$, $-$)
(4) **Si** Atributos = $\emptyset$ **Return** ($raíz$, l)
(l es el valor de la clase mayoritaria de $Clases$ en $Ejemplos$)

Si ninguna de las condiciones anteriores se cumple
Inicio
(1) Seleccionar el atributo A en $Atrubutos$ con mayor $Ganancia(Ejemplos, A)$
(2) El atributo de decisión para $raíz$ es A.
(3) Para cada posible valor $v_i$de A.
(3.1) Añadir una rama a $raíz$ con valor v_i
(3.2) $Ejemplos\_v_i$ es el subconjunto de $Ejemplos$ con valor v_i para A
(3.3) **Si** $Ejemplos\_v_i = \emptyset$
Entonces añadir un nodo (n, l) a partir de la rama creada
(l es el valor de la clase mayoritaria de $Clases$ en $Ejemplos$)
Si no añadir a la rama creada el subárbol
ID3$(Ejemplos\_v_i,\ Clases,\ Atributos - \{A\})$
Fin

Figura 5. Pseudocódigo para el algoritmo ID3.

2.3. REDES PARCIALMENTE CONECTADAS.

En muchos casos una red MLP puede contener conexiones innecesarias, lo cual induce a cuestionarse la complejidad de las redes, además de provocar un tiempo de entrenamiento lento, especialmente para redes de gran tamaño. La reducción de conexiones o neuronas puede lograrse descartando los pesos que menos contribuyen a las salidas de la red. [2005, Kang et al.]

Existen varios trabajos relacionados al problema de reducción de conexiones en redes MLP's. En el artículo de [1997, Elizondo et al.], se muestran tres grupos de métodos para la reducción de conexiones y obtención de una red parcialmente conectada. Estos métodos son ontogénicos, no ontogénicos e híbridos. Los métodos ontogénicos reducen el número de conexiones en la topología de la red durante el proceso de aprendizaje; en los no ontogénicos se hace un análisis de los datos para eliminar conexiones antes de entrenar la red y los métodos híbridos usan una combinación de ambos. Por ejemplo, en [2005, Kang et al.] realizan la eliminación de conexiones identificando el tipo de entrada, determinando si el atributo es *acoplado,* esto es, que está asociado multiplicativamente con otro atributo o en si sólo está asociado aditivamente, entonces el atributo está *desacoplado*. Esta identificación del tipo de las entradas se hace durante el proceso de apren-

dizaje, si al menos una de las entradas de la red es de tipo desacoplada, entonces es factible modelar una red parcialmente conectada.

En [1990, Sethi] se propone una red MLP parcialmente conectada a partir de la construcción de un árbol de decisión binario, donde la red consta de dos capas ocultas. El autor llamó a este tipo de redes *Entropy nets.* Debido a que el enfoque que utiliza es similar al utilizado en esta tesis, en el siguiente apartado se explica detalladamente la metodología que Sethi propuso para la construcción de una red MLP parcialmente conectada.

2.3.1. ENTROPY NETS

La topología de este tipo de redes consta de dos capas ocultas. La primera capa es llamada *partitioning layer,* la segunda *ANDing layer* y la capa de salida es llamada *ORing layer*. Las capas atañen sus nombres a la analogía que hace el autor al proceso de clasificar una entrada desconocida en un árbol de decisión. Cuando un objeto desconocido es presentado a un árbol de decisión se recorre el árbol del nodo raíz a una de sus hojas que determina la clase del objeto. Para llegar a un nodo hoja, el objeto debe cumplir con todas las condiciones de alguna de las reglas que se obtuvieron en el proceso de aprendizaje que tienen la forma (7) lo que implica operaciones AND; si el objeto no cumple una regla, entonces cumple otra, lo que implicaría al menos una operación OR. Observemos que las operaciones AND particionan el espacio de características y las operaciones OR el espacio de decisión.

Las reglas para la construcción de una *Entropy net* dado un árbol de decisión binario son las siguientes:

- El número de neuronas en la primera capa oculta es igual al número de nodos internos del árbol de decisión. Cada una de estas neuronas implementa una de las operaciones de decisión de los nodos internos. Esta capa es llamada *partitioning layer*.
- Todos los nodos hoja tienen una correspondiente neurona en la segunda capa oculta donde la *ANDing layer* es implementada.
- El número de neuronas en la capa de salida es igual al número de distintas clases. Esta capa corresponde a la *ORing layer.*
- Las conexiones entre las neuronas de la *ANDing layer* y la *ORing layer* se realizan de manera que cada neurona con su clase asociada a un nodo hoja de la *ANDing layer* se conecte a la neurona en la *ORing layer* que tenga la misma clase.

- Las conexiones entre las neuronas de la *partitioning layer* y las neuronas de la *ANDing layer* implementan la jerarquía del árbol.

Para ejemplificar la definición de la topología de una red MLP tipo *Entropy net* considérese el árbol de decisión, la figura 6(a) correspondiente al problema XOR y su respectiva red MLP parcialmente conectada se muestra en la figura 6(b). El problema XOR tiene dos clases; *True* y *False*, donde los objetos tienen dos atributos binarios; x e y, un objeto pertenece a la clase *True* si el valor de sus atributos es distinto, en caso contrario pertenece a la clase *False*.

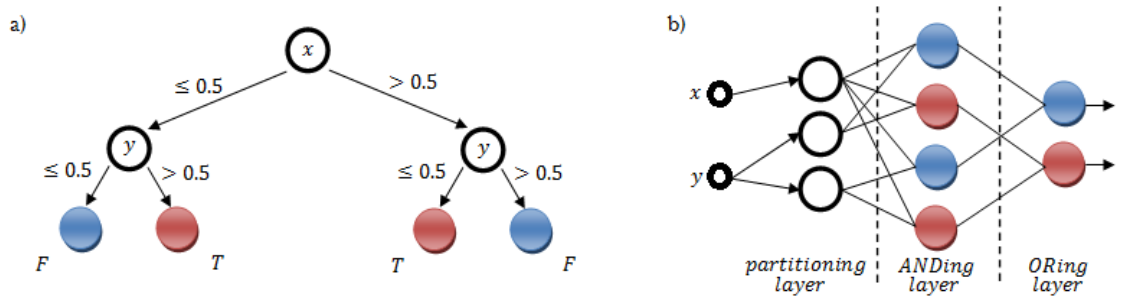

Figura 6. a) Árbol de decisión correspondiente al problema XOR. **b)** Entropy Net asociada al problema XOR.

Se puede observar que en este tipo de problemas se conoce la salida deseada de las neuronas en la segunda capa oculta, por lo que el autor propuso su propio algoritmo de aprendizaje para entrenarlas; sin embargo, también pueden ser entrenadas con el algoritmo *back-propagation.*

CAPÍTULO 3

TOPOLOGÍA DE UNA RED MLP BASADA EN GANANCIA DE INFORMACIÓN

En este capítulo se define el modelo y las reglas para construir una red MLP con la topología propuesta en esta tesis. La propuesta se basa en la métrica *ganancia de información* utilizada en la construcción de árboles de decisión (ver sección 2.2). A las redes construidas con ésta topología las hemos llamado *IG nets* (*Information Gain nets*). Además, se describe la implementación y los requerimientos para la construcción de una *IG net*.

3.1. DESCRIPCIÓN DEL MODELO

Los valores de los atributos de los objetos determinan la clase a la que estos pertenecen. La importancia de un atributo en una red neuronal artificial está asociada a los pesos que la red le asigna a las entradas de cada neurona en donde está presente el atributo de manera directa (en la primera capa oculta) o indirectamente (en el resto de las capas). En un árbol de decisión, la importancia que proporciona un atributo para discriminar las clases, está determinada por la ganancia de información del atributo, la cual se describió en el apartado 2.2.1.

Las reglas de la forma (7) del capítulo anterior, particionan el espacio de decisión ya que permiten determinar a qué clase pertenece un objeto de acuerdo a sus características. Estas reglas de clasificación tienen condiciones que deben cumplir los atributos de un objeto para determinar si pertenece a la clase asociada a una regla.

Cuando un nodo p en un árbol de decisión tiene m nodos hoja, los atributos asociados a los nodos involucrados en el camino para llegar a cada una de estas hojas son los mismos, partiendo desde el nodo raíz r. Entonces se tienen m reglas de decisión diferenciándose entre sí en la última condición. Esto nos lleva a la observación de que los objetos que recaen en esas m hojas, determinan su clase con los mismos atributos, y que pueden ser descritos en el mismo espacio dimensional. De lo anterior se puede plantear el utilizar las neuronas de la primera capa oculta en una red MLP entrenada con el algoritmo *back-propagation* para encontrar hiperplanos de separación en los espacios especificados por las reglas de un árbol de decisión, y las salidas de estas neuronas alimenten a las neuronas de la capa de salida.

En este trabajo, se presenta una manera de obtener una red MLP a partir de un árbol de decisión. A continuación se especifican las reglas que definen la topología de la red basándose en la siguiente notación:

Sea A un árbol de decisión construido con el conjunto de entrenamiento T. Se tiene:

$M_p =$ Número de nodos que son padres de al menos un nodo hoja en A. En caso de que el nodo raíz sea padre de al menos un nodo hoja, éste no se toma en cuenta.

$p_i =$ El $i - ésimo$ nodo padre de hojas donde $j = 1 \dots M_p$.

$m_i =$ Número de hojas de p_i.

$k_{ij} =$ Clase asociada a la $j - ésima$ hoja de p_i, donde los posibles valores de j están entre 1 y m_i.

Las reglas de definición de la topología de la red MLP se presentan a continuación, asociándoles un identificador para su posterior explicación:

R1. El número de neuronas en la capa de entrada es igual al número de atributos involucrados en A.

R2. El número de neuronas en la capa de salida, se define de la misma manera que en la topología estándar de una red MLP. Recordemos que cada neurona en esta capa corresponde a una clase.

R3. El número de neuronas en la capa oculta es igual a M_p. Cada una de estas neuronas g_i tienen un correspondiente nodo p_i.

R4. Las entradas de cada neurona g_i son los atributos involucrados en el camino para llegar desde el nodo raíz de A hacia p_i.

R5. Para el nodo raíz r de A con el atributo x_j asociado y con n nodos hoja, si $n > 0$ denotemos como k_i la clase de la $i - ésima$ hoja de r para $i = 1 \dots n$. Entonces la neurona en la capa de entrada correspondiente a x_j se conecta directamente a los nodos en la capa de salida con clase k_i.

R6. La salida de cada neurona g_i alimenta a las neuronas en la capa de salida con clase k_{ij} para cada $j = 1 \dots m_i$.

Cuando uno o más atributos no estén involucrados en las reglas de decisión del árbol, entonces no se tomarán en cuenta en las entradas de la red, debido a que ninguno de los espacios definidos por el árbol estarán determinados por dicho(s) atributo(s), lo que justifica a R1. Dado que *A* describe a todo *T*, existe al menos una hoja asociada a cada clase en T, lo cual explica R2.

Es importante resaltar que una red MLP con una sola capa oculta es capaz de representar un problema tipo IOM. Esto se prueba a través del *teorema universal de aproximación* [1988, Cybenko]. Por esta razón, en la topología de las *IG nets* se define sólo una capa oculta correspondiente a la regla R3.

La topología de las *IG nets* intenta dotar a la red con el número de neuronas ocultas adecuado a los objetos en el conjunto de entrenamiento. A fin de conseguirlo, a cada espacio definido por el árbol de decisión se le asigna una neurona escondida, proporcionándole a cada neurona las *habilidades* necesarias para generar hiperplanos en dicho espacio. Esta habilidad se obtiene teniendo como entradas a los atributos que describen el espacio, lo que explica la regla R4.

Cuando el nodo raíz es padre de al menos una hoja, se define un espacio de decisión que depende del atributo asociado al nodo raíz, por lo tanto la neurona asociada a este nodo tiene como entrada un solo atributo. Esta tarea puede asignarse directamente a las neuronas en la capa de salida que correspondan a las clases involucradas en dicho espacio, obteniendo así la regla R5.

Las neuronas en la capa de salida se encargan de determinar qué neurona en la capa oculta discrimina mejor la pertenencia de un objeto a una clase. Para lograrlo, las salidas de las neuronas ocultas alimentan a las neuronas en la capa de salida con la(s) clase(s) que corresponde(n) a su espacio de decisión asignado, lo cual es expresado en la regla R6.

Por ejemplo, para el árbol de decisión de la figura 3 y mostrado nuevamente en la fi gura 7(a), se tiene la *IG net* mostrada en la figura 7(b), la cual tiene dos neuronas ocultas a pesar de existir tres nodos padres de hojas en el árbol de decisión, esto es debido a la regla R5. En la figura 7(c) se muestra a manera de comparación la *Entropy net* asociada al mismo árbol, donde puede apreciarse que el número de neuronas ocultas es igual al número de nodos en el árbol de decisión.

Nótese que en la topología de las *IG nets* el número de neuronas en la capa oculta, depende directamente del árbol a partir del cual se construye. Cabe recordar que los criterios de poda utilizados en un árbol de decisión, son utilizados para proporcionar una mayor generalización. Sin embargo, para la construcción de una *IG net*, se considera pertinente no utilizar ningún criterio de

poda en la construcción del árbol, para que las neuronas en la capa oculta consideren todos los subespacios posibles, dejándole la actividad de generalización a la red.

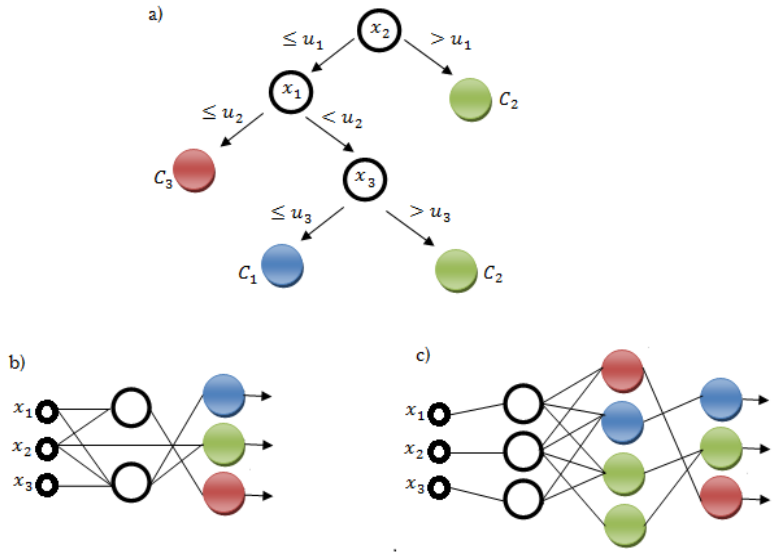

Figura 7. a) Árbol de decisión mostrado en la figura 3 para un problema de 3 clases y tres atributos. **b)** IG net asociada a nuestra propuesta. **c)** Entropy net asociada a a). Los colores en los nodos del árbol y la red representan las clases.

3.2. IMPLEMENTACIÓN

Para la implementación de las *IG nets* se pueden destacar tres requerimientos funcionales a partir de la suposición de que se dispone de los datos de entrenamiento:

1. **Construcción del árbol de decisión.** Generar un árbol de decisión a partir de los datos de entrenamiento almacenándolo en una estructura de datos para disponer de la información necesaria.
2. **Extracción de información.** De la estructura obtenida en 1 se extrae la información necesaria para la construcción de una red *IG net*, como lo es el número de nodos que son padres, los nodos en el camino para llegar a ellos, así como los nodos hoja de cada uno de ellos.
3. **Construcción de la IG net.** Proporcionarle a la red la información obtenida en 2, para obtener una red con la topología propuesta.

Para implementar estos requerimientos se utilizó el leguaje de programación Java haciendo uso de NetBeans IDE [2012, Tulach] y una librería hecha en Java llamada Weka [2009, Hall et al.], estas aplicaciones cuentan con licencia GNU. NetBeans IDE es un ambiente de desarrollo integrado para programadores que proporciona las herramientas necesarias para crear aplicaciones pro-

fesionales de escritorio, empresariales, web y móviles con la plataforma de Java, asi como con C/C++, PHP, JavaScript y Groovy. Weka tiene implementados diversos algoritmos para la minería de datos, incluyendo entre otros componentes, métodos de clasificación y clustering, herramientas para el pre-procesamiento de los datos como lo son filtros para los atributos, además de distintos métodos para la selección de atributos. Dentro de los algoritmos de clasificación que tiene implementados, se encuentran árboles de decisión y redes MLP.

La librería Weka está compuesta por distintas clases que permiten instanciar objetos para entrenamiento y/o prueba (*Instances.java*, *Instance.java*) y sus atributos (*Attribute.java*), así como las clases que permiten construir un clasificador que heredan de la clase *AbstractClassifier.java* que, como lo indica su nombre, es una clase abstracta que a su vez es una implementación de la interfaz *Classifier.java*. Esta interfaz obliga a cada clasificador a tener, entre otros métodos, uno llamado *buildClassifier* que recibe como parámetro el conjunto de objetos de entrenamiento. Este método inicializa todas las variables mediante parámetros que proporciona el usuario y construye un clasificador, adecuado a los datos del conjunto entrenamiento (ver figura 8).

3.2.1. CONSTRUCCIÓN DEL ÁRBOL DE DECISIÓN

Se implementó el software necesario para construir redes con modelo *IG net* a partir de un árbol de decisión creado con el algoritmo C4.5. Este algoritmo en la librería Weka se encuentra implementado en la clase *j48.java*, donde una vez construido el árbol mediante *buildClassifier*, cada nodo es del tipo *ClassifierTree* (ver apéndice A), los hijos de cada nodo se encuentran en un arreglo de nodos del mismo tipo, formando la estructura del árbol recursivamente. De acuerdo a la descripción del modelo de las *IG nets*, es necesario conocer información para cada nodo que no está contemplada en la librería de Weka, por lo que se realizaron las siguientes modificaciones a la clase *ClassifierTree.java*:

- Se agregó un atributo que indica el número de hojas, con la finalidad de conocer el número de nodos que son padres de nodos hoja. Un nodo es padre de una hoja si este atributo es mayor a 0. Además este atributo permite conocer el tamaño del arreglo que tiene a los nodos hoja.
- Para determinar el tamaño del camino para llegar del nodo raíz a un nodo padre de hoja es necesario conocer en qué nivel se encuentra cada nodo padre de hoja. Es por esto que se añadió un atributo que indica el nivel en que se encuentra cada nodo.

Una vez construido el árbol con la información adicional, se pueden crear métodos que accedan a la estructura para extraer la información que se le proporcionará a la red MLP para tener la topología *IG net*.

En la figura 8 se muestra un diagrama de clases que permite dar una idea general de la relación entre las clases utilizadas de la librería Weka, la documentación de la librería se puede consultar en [2009, Hall et al.].

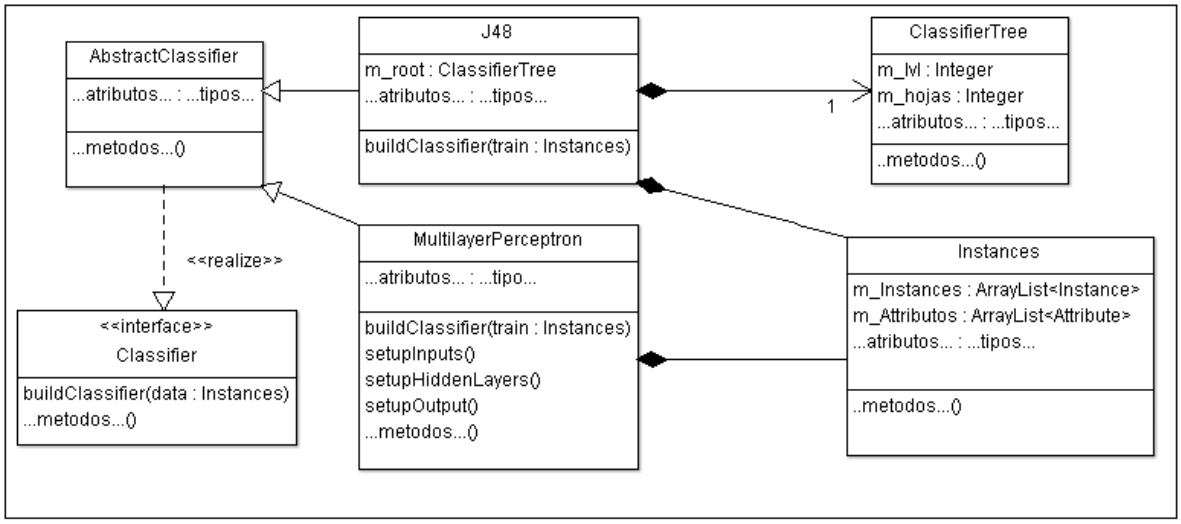

Figura 8. Diagrama de clase con las relaciones entre las clases utilizadas de la librería Weka para la implementación del modelo *IG net*.

3.2.2. EXTRACCIÓN DE INFORMACIÓN

La información del árbol de decisión necesaria para la construcción de una red MLP con la topología propuesta es la siguiente:

- Número de nodos que son padres de nodos hoja.
- El atributo de cada nodo en el camino para llegar de la raíz a cada nodo padre de hojas.
- La clase asociada a cada nodo hoja de cada nodo padre.

Para representar esta información considérese la figura 9. La figura 9 (a) corresponde al árbol de decisión mostrado en la figura 3. La figura 9 (b) corresponde a la estructura de datos asociada a 9(a).

En la figura 9(b), se observa que la estructura consiste en un arreglo con tamaño igual al número de nodos padre (en negro en la figura), y para cada elemento del vector se tienen dos arreglos, uno corresponde a los atributos involucrados en el camino para llegar al nodo padre de hoja, y el otro a las clases de cada una de sus hojas. El orden en que se obtiene la información para cada uno de los nodos padres de hoja, se obtiene al recorrer el árbol en preorden.

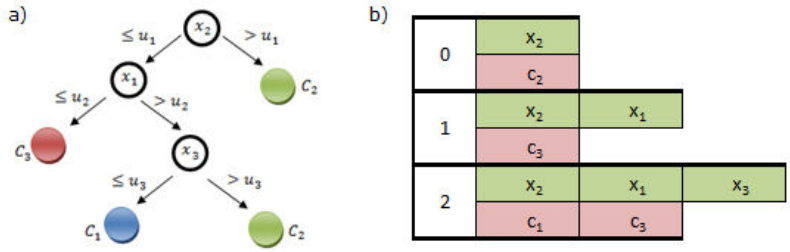

Figura 9. a) Árbol de decisión mostrado en la figura 3. **b)** Estructura de los datos extraidos del árbol. Las $\boldsymbol{X_i}$ son los atributos y $\boldsymbol{C_i}$ corresponde a las clases.

3.2.3. CONSTRUCCIÓN DE LA IG NET

Una *IG net* es una red MLP construida bajo las reglas descritas en la sección 3.1. La librería Weka tiene implementado un algoritmo para construir una red MLP con topología estándar en la clase *MultilayerPerceptron.java.* Esta clase, al heredar de *AbstractClassifier.java,* tiene implementado el método *buildClassifier*, donde crea primero las neuronas de entrada con el método *setupInputs*, después con *setupOutputs* las de salida y por último las ocultas con *setupHiddenLayer*. Las conexiones entre neuronas las realiza conforme se crean las neuronas ocultas; estas conexiones son manejadas en la clase *NeuralConection.java*. Dos neuronas son conectadas con el método *connect,* el cual recibe dos parámetros; la neurona que alimenta y la neurona alimentada.

Para la obtención de una *IG net*, se crean las neuronas de entrada y de salida de la misma manera que en la topología estándar; para la obtención de las neuronas ocultas y las conexiones entre neuronas se utiliza la información extraída del árbol de decisión. Por ejemplo, para la estructura obtenida en la figura 9(b) la idea se expresa en la figura 10(a), donde no se toma en cuenta la regla R5 descrita en la sección 3.1, la cual indica que, en caso de que el nodo raíz del árbol sea padre de al menos un nodo hoja, solo existe un atributo involucrado, por lo que el nodo en la capa de entrada de la red que corresponde al atributo se conecta directamente a los nodos con las clases involucradas. El resultado de aplicar esta regla es mostrado en la figura 10(b).

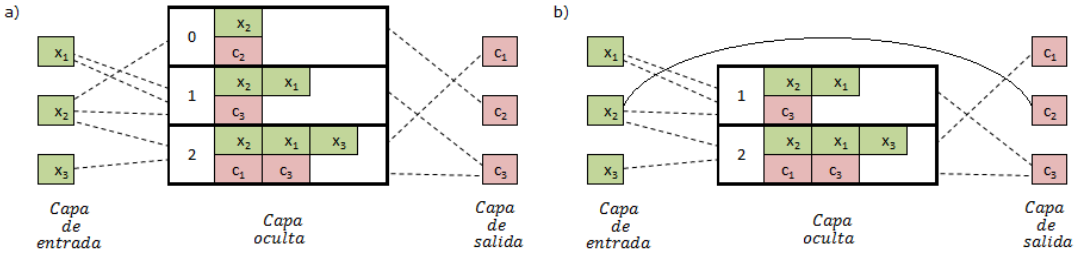

Figura 10. a) Obtención de capas ocultas utilizando información del árbol. **b)** Regla R5 aplicada a a).

Una vez construida la red MLP con la topología *IG net,* se entrena con el algoritmo backpropagation para poder así evaluar su desempeño.

En la figura 11 se muestra la interfaz gráfica del sistema desarrollado que tiene implementado los requerimientos funcionales para la construcción de redes con modelo *IG net*.

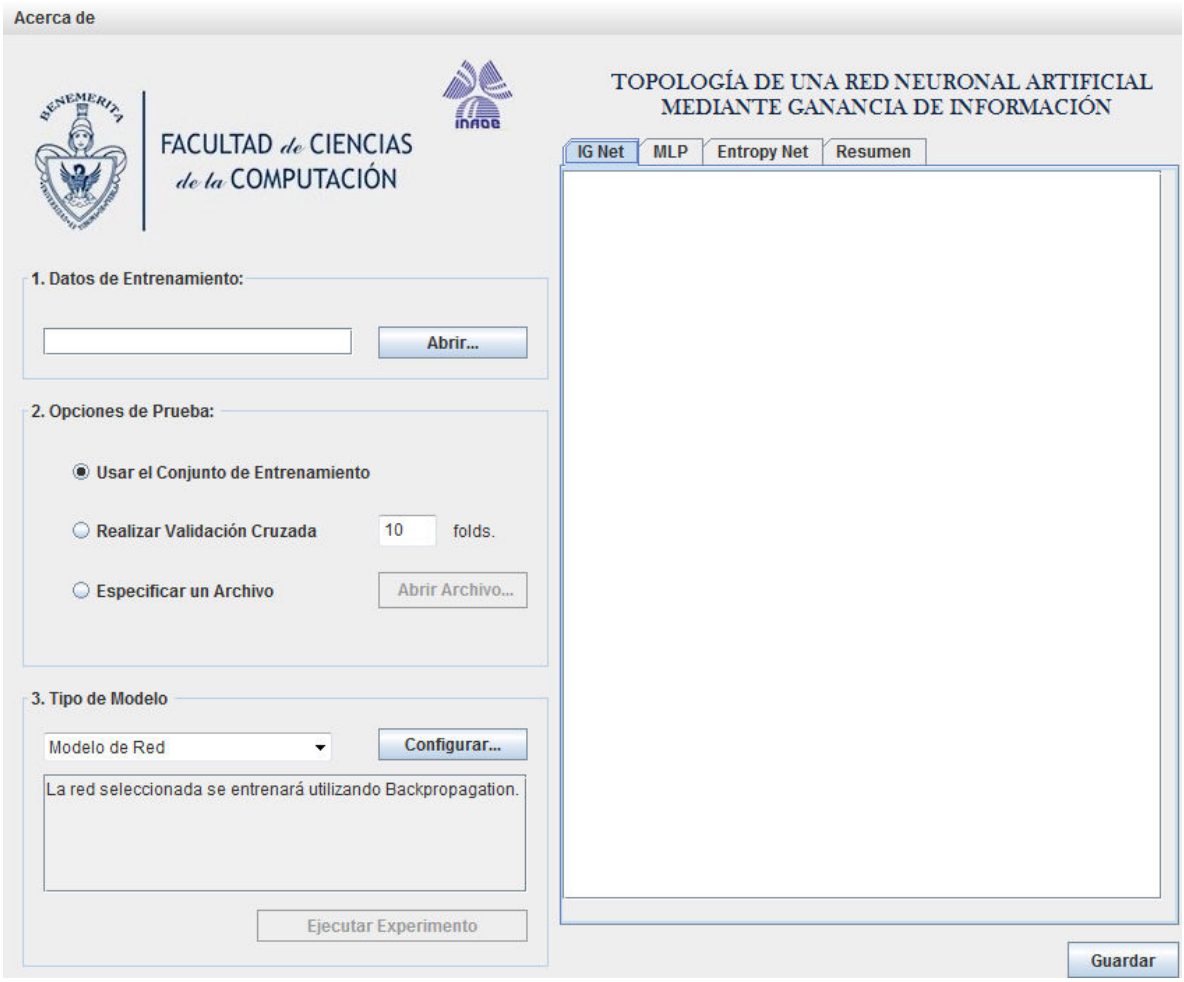

Figura 11. Interfaz gráfica del sistema que implementa el modelo de las *IG nets*.

CAPÍTULO 4

PRUEBAS Y RESULTADOS

Este capítulo describe las pruebas realizadas para comparar el desempeño de la topología propuesta respecto a la estándar y a la propuesta en [1990, Sethi], así como la metodología seguida para probar estadísticamente los resultados. Además se hace un análisis de los resultados obtenidos para concluir respecto al desempeño de las *IG nets*.

4.1. DESCRIPCIÓN DEL EXPERIMENTO

Para comparar el desempeño de las *IG nets*, se implementó el sistema *PCN-Tree (Partial Connected Nets - Tree)* que permite; dado un conjunto de datos de entrenamiento, construir tres modelos de redes MLP para su posterior evaluación. Los posibles modelos implementados en el sistema corresponden a las *IG nets*, *Entropy Nets* y las redes MLP totalmente conectadas y pueden ser evaluados ya sea con los mismos datos de entrenamiento, un conjunto de prueba específico ó realizando validación cruzada. Se puede consultar la descripción de escenarios, la verificación de los modelos de red MLP implementados, así como el manual de usuario del sistema en los apéndices B, C y D respectivamente.

El experimento consiste en dos fases, la primera fase tiene la finalidad de verificar que al evaluar el desempeño de las *IG nets* alcancen un porcentaje aceptable de objetos clasificados correctamente; por aceptable entenderemos que el porcentaje de clasificación es igual o mejor respecto a los otros modelos. En la segunda fase, se evalúan los tiempos de entrenamiento, el número de conexiones y el número de neuronas ocultas de cada modelo.

La validez de los resultados se prueba haciendo uso de una prueba de significancia estadística conocida como *Wilcoxon Signed-Ranks test (WSR)* debido a que permite la comparación de dos o más clasificadores sobre múltiples conjuntos de datos [2006, Demšar], [1997, Dietterich] y se adapta a las necesidades de las pruebas realizadas en ambas fases. Las pruebas se realizaron con un nivel de significancia de 0.5 y los datos con los que se entrenó y evaluó el desempeño de cada modelo fueron obtenidos de *UCI Machine Learning Repository* [2010, Frank et al.], un repositorio disponible al público. Una breve descripción de los datos se muestra en la tabla 1.

Para cada conjunto de datos se construyeron cuatro modelos de redes descritos a continuación:

- ***IG net.*** Red MLP con la topología propuesta en esta tesis. El árbol de decisión para construir la red no fue podado y se discretizó cada atributo previamente a la construcción.

- ***Entropy net*.** Red MLP con la topología propuesta en [1990,Sethi]. El árbol de decisión para construir la red fue podado.
- **MLP-IG.** Red MLP con topología estándar, una sola capa oculta con el mismo número de neuronas ocultas que una *IG net*.
- **MLP-Entropy.** Red MLP con topología estándar con el mismo número de capas y neuronas ocultas correspondientes a una *Entropy net*.

Los parámetros utilizados para cada red fueron 0.3 para la tasa de aprendizaje, 0.2 para el momentum y $1{,}000$ épocas.

Tabla 1. Descripción de los conjuntos de datos utilizados para los experimentos.

Nombre	Objetos	Atributos	Clases
Iris	150	4	3
Wine	178	13	3
Connectionist Bench	208	60	2
SPECT Heart	267	44	2
Ionosphere	351	34	2
Breast Cancer	699	9	2
Pima Indians	768	8	2
Statlog	2,310	19	7
Spambase	4,542	57	2
Page Blocks	5,473	10	5

4.1.1. PRIMERA FASE

Para cada conjunto de datos se construyó cada modelo realizando validación cruzada con 10 *folds*, esto es, se divide el conjunto de datos en 10 partes aproximadamente del mismo tamaño, 9 son utilizadas para entrenar la red y la parte restante es utilizada para evaluar la red. Este proceso se repite hasta haber utilizado cada una de las 10 partes para evaluar la red y utilizando el resto en la etapa de entrenamiento. Se reporta el porcentaje de objetos clasificados correctamente en cada *fold* para cada conjunto de datos. El resultado de este proceso para el conjunto de datos Iris es mostrado en la tabla 2.

Se utiliza la prueba WSR para comparar el desempeño obtenido entre la *IG net* y cada uno de los otros modelos para cada uno de los conjuntos de datos, y así determinar si existe diferencia significativa. Esta prueba consiste en obtener la diferencia entre los porcentajes de clasificación de los modelos a comparar. Siempre que la diferencia sea distinta de cero, se asigna un *rank* al valor absoluto de las diferencias, asignando 1 al menor valor, 2 al segundo, hasta que cada dife-

rencia tenga un *rank* asignado, en caso de empates se asigna la media de los *ranks* implicados en el empate. Se realiza la suma de los *ranks* que provienen de una diferencia negativa y la suma de los que provienen de una diferencia positiva, el mínimo valor de estas sumas, será el valor del estadístico de prueba T, que se compara con el valor obtenido de la tabla de valores críticos para T en la tabla para la prueba WSR (para n igual al número de diferencias distintas de 0). Si el estadístico de prueba es mayor al valor crítico, quiere decir que hay diferencia significativa en los valores comparados.

Tabla 2. Porcentaje de clasificación al evaluar cuatro modelos de redes MLP utilizando validación cruzada con 10 *folds* para el conjunto de datos Iris.

FOLD	IG net	Entropy Net	MLP - IG	MLP – Entropy
1	86.67	93.33	86.67	86.67
2	93.33	86.67	100.00	100.00
3	93.33	93.33	93.33	93.33
4	100.00	100.00	100.00	100.00
5	100.00	93.33	100.00	100.00
6	100.00	100.00	100.00	100.00
7	93.33	93.33	73.33	73.33
8	100.00	100.00	100.00	100.00
9	100.00	93.33	100.00	100.00
10	100.00	100.00	100.00	100.00

Al realizar la prueba WSR, sólo se encontró diferencia significativa entre los porcentajes de clasificación de la *IG net* y la red MLP-IG obtenidas con el conjunto de datos *Spambase*, donde fue mejor el desempeño de la *IG net*. En la tabla 3 se muestra el promedio de los porcentajes de clasificación de los 10 *folds* realizados en la validación cruzada (ver figura 12) y su respectiva desviación estándar para cada conjunto de datos.

Tabla 3. Medias y desviaciones estándar de los porcentajes de clasificación obtenidos de realizar validación cruzada con 10 *folds* para evaluar cuatro modelos de redes MLP con los conjuntos de datos descritos en la tabla 1.

Conjunto de Datos	IG net		Entropy Net		MLP – IG		MLP – Entropy	
	%	σ	%	σ	%	σ	%	σ
Wine	**93.89**	6.11	**95.49**	3.54	**97.16**	3.98	**97.75**	3.92
Ionosphere	**91.44**	4.27	**90.31**	5.08	**89.75**	4.89	**90.60**	4.68
Iris	**96.67**	4.71	**95.33**	4.50	**95.33**	8.92	**95.33**	8.92
Pima Indians	**74.87**	4.77	**77.08**	4.91	**73.19**	4.70	**73.70**	4.84
Breast Cancer	**95.84**	2.40	**95.70**	2.15	**95.84**	2.75	**95.27**	2.63
Spambase	**92.83**	1.52	**92.83**	1.85	**90.04**	4.57	**92.18**	1.16
SPECT Heart	**80.17**	6.33	**81.68**	5.28	**78.65**	7.01	**77.14**	6.61
Connectionist Bench	**79.31**	6.43	**77.45**	8.36	**83.62**	7.60	**79.31**	10.34
Statlog	**96.80**	1.56	**97.10**	0.71	**97.40**	1.50	**97.45**	0.75
Page Blocks	**96.55**	0.72	**96.67**	0.73	**96.53**	0.75	**96.64**	0.77

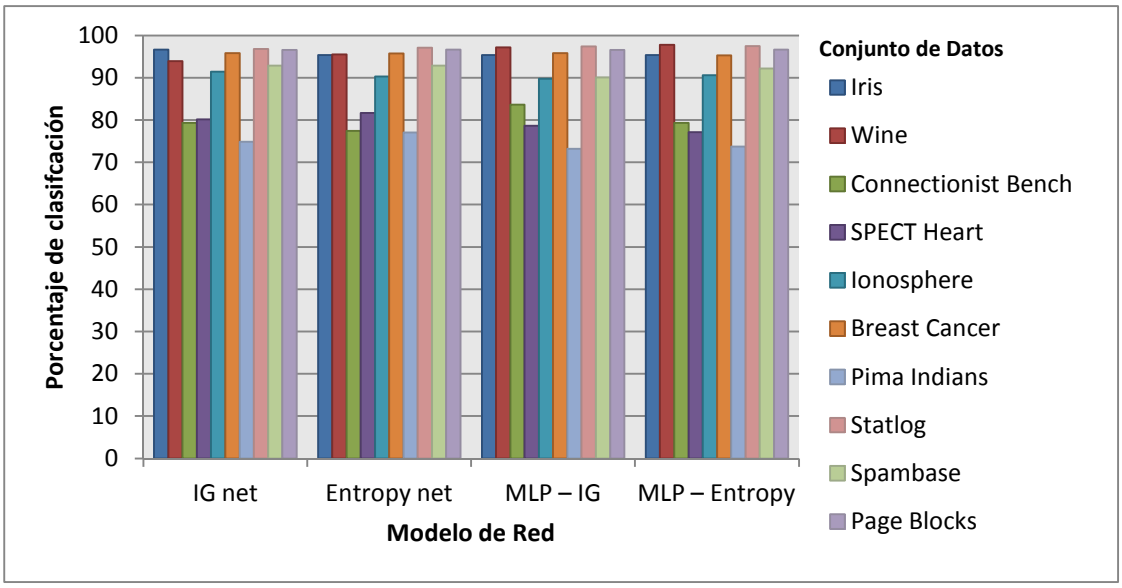

Figura 12. Promedio del porcentaje de clasificación al realizar validación cruzada con 10 *folds*.

4.1.2. SEGUNDA FASE

Para determinar si existe diferencia significativa en el tiempo de entrenamiento y el número de conexiones entre los modelos, se realizó también la prueba WSR utilizando los datos mostrados en la tabla 4 y representados en las figuras 13 y 14. En la figura 13 se muestra una gráfica de barras de los tiempos de entrenamiento y en la figura 14 la gráfica para el número de conexiones. Se comparó los tiempos obtenidos al entrenar las *IG nets* construidas con los diez conjuntos de datos respecto a los obtenidos en cada uno de los otros tres modelos. En la tabla 5 se muestran los resultados de las comparaciones realizadas.

Tabla 4. Tiempo de entrenamiento (t) medido en segundos y número de conexiones (c) para cada uno de los cuatro modelos de red MLP construidos con los conjuntos de datos de la tabla 1.

Conjunto de Datos	IG net		Entropy net		MLP – IG		MLP – Entropy	
	t	c	t	c	t	c	t	c
Iris	**0.62**	21	**0.61**	16	**0.73**	35	**0.87**	36
Wine	**0.58**	14	**0.96**	21	**0.96**	48	**1.77**	87
Connectionist Bench	**2.85**	96	**2.39**	63	**12.49**	930	**10.75**	732
SPECT Heart	**6.54**	251	**4.79**	121	**18.27**	1,058	**17.52**	932
Ionosphere	**3.66**	68	**3.39**	57	**11.1**	468	**9.75**	362
Breast Cancer	**8.77**	85	**14.28**	113	**11.4**	187	**24.38**	407
Pima Indians	**24.89**	288	**41.02**	355	**33.53**	460	**114.88**	1,952
Statlog	**88.75**	281	**142.31**	445	**218.1**	1,300	**513.68**	3,017
Spambase	**1,636.23**	3,203	**1,530.60**	2,421	**3,499.69**	12,036	**18,252.56**	36,351
Page Blocks	**333.17**	499	**475.92**	133	**560.4**	1,275	**2,156.26**	5,417

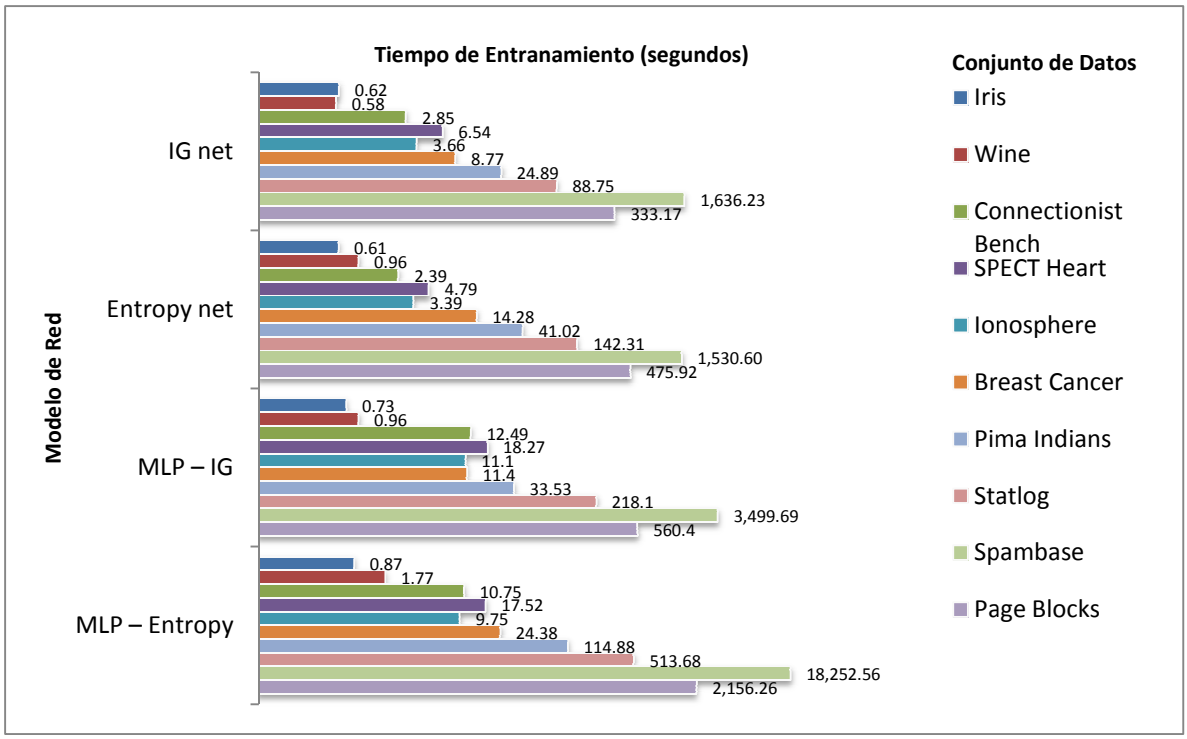

Figura 13. Gráfica de barras para el tiempo de entrenamiento de cada modelo de red para cada conjunto de datos.

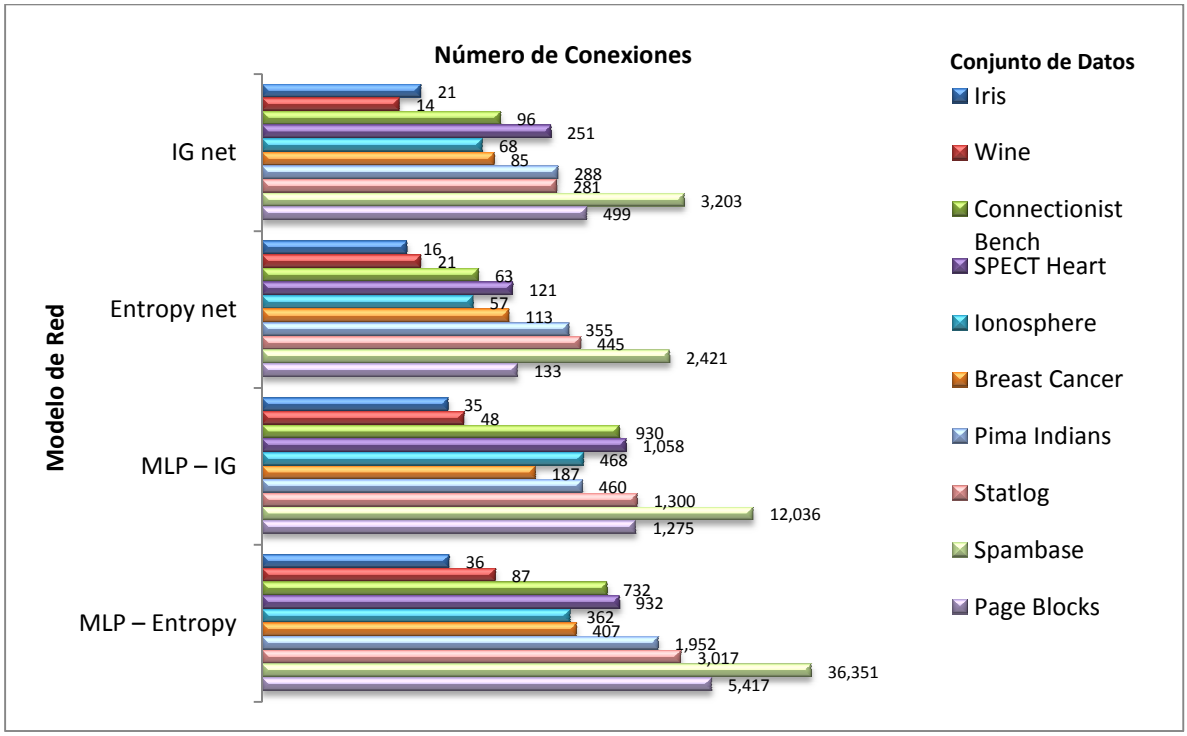

Figura 14. Gráfica de barras para el número de conexiones de cada modelo de red para cada conjunto de datos.

Tabla 5. Comparación del número de conexiones y tiempo de entrenamiento utilizando la prueba WRS. El símbolo '*' representa qué modelo fue significativamente mejor.

Conexiones		Tiempo (segundos)	
IG net	Entropy net	IG net	Entropy net
IG net*	MLP-IG	IG net*	MLP-IG
IG net*	MLP-E	IG net*	MLP-Entropy

Para comparar el número de neuronas ocultas, notemos que los modelos *IG net* y MLP-IG tienen el mismo número de neuronas, mismo caso de los modelos *Entropy net* y MLP-Entropy, es por esto que sólo se comparó el número de neuronas entre los modelos *IG net* y *Entropy net*. En la comparación realizada con la misma prueba estadística se encontró que la diferencia fue significativa, tomando ventaja el modelo *IG net*. En la figura 15 se muestra el número de neuronas ocultas en cada uno de estos dos modelos para cada conjunto de datos, donde puede apreciarse que en cada uno de los conjuntos de datos el número de neuronas ocultas es menor para el modelo *IG net*.

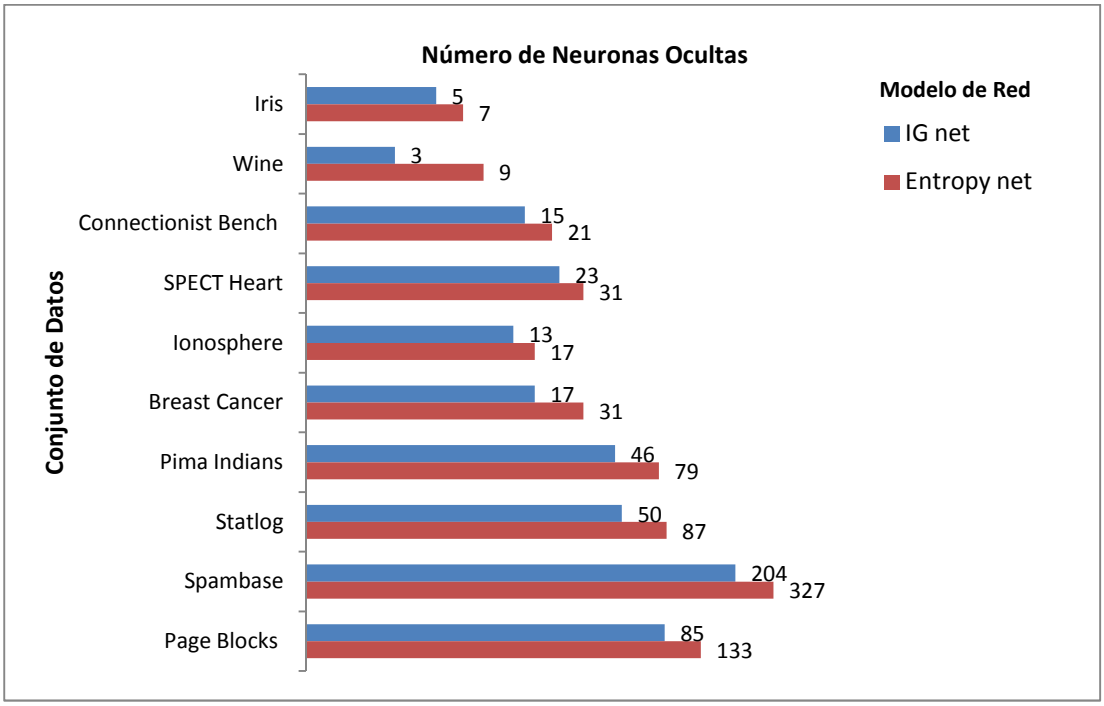

Figura 15. Número de neuronas ocultas en las redes construidas con modelo *IG net* y *Entropy net* para los conjuntos de datos de la tabla 1.

4.2. ANÁLISIS DE RESULTADOS

El objetivo de las redes MLP parcialmente conectadas es reducir costos computacionales respecto a las redes totalmente conectadas, dicho objetivo debe ser alcanzado sin tener pérdida significativa en el porcentaje de clasificación de la red. En la primera fase de experimentos se encontró que el porcentaje de clasificación de las *IG net* respecto a los otros tres modelos, no mostró diferencias significativas. Este resultado indica que utilizar cualquiera de estos cuatro modelos de redes MLP proporciona un porcentaje de clasificación con una diferencia no significativa.

Dado que se tiene un porcentaje de clasificación aceptable para el modelo *IG net*, la fase dos de experimentos resulta decisiva; determinar qué modelo utilizar radica en el costo computacional necesario en cada modelo. En cuanto al tiempo de entrenamiento y el número de conexiones, se encontró que el modelo *IG net* es significativamente mejor a los modelos construidos con topología estándar. Esta ventaja no se obtiene con el modelo *Entropy net*, debido a que no hubo diferencia significativa para estos valores. Sin embargo, las *IG nets* resultaron significativamente mejores en el número de neuronas ocultas, ventaja que favorece en cuanto a la cantidad de memoria necesaria.

CAPÍTULO 5

CONCLUSIONES Y TRABAJO FUTURO

En este capítulo se concluye acerca de las redes con la topología propuesta en esta tesis definida mediante ganancia de información, incluyendo una sección del trabajo futuro a desarrollar para complementar el trabajo realizado.

Las redes MLP construidas bajo el modelo de las *IG nets* alcanzan el objetivo de las redes parcialmente conectadas, ya que reducen costos computacionales en cuanto al tiempo de entrenamiento y cantidad de memoria necesaria al eliminar conexiones innecesarias, sin tener pérdida significativa en el porcentaje de clasificación obtenido al evaluar la red.

El uso de la topología propuesta en esta tesis para un problema IOM específico, elimina el problema de decidir el número de capas y neuronas ocultas que debe tener la red, ya que el número de neuronas se determina mediante la construcción del árbol de decisión. Esto evita entrenar y evaluar redes experimentando con distintos números de neuronas y capas ocultas.

La capacidad discriminativa de cada atributo de los objetos en un problema IOM determinada mediante ganancia de información para decidir la clase a la que pertenece un objeto, permite reducir conexiones innecesarias en una red MLP totalmente conectada.

5.1. TRABAJO FUTURO

El árbol de decisión que permite la construcción de una *IG net* para un problema IOM específico, determina el número de neuronas ocultas y las conexiones que ésta tendrá, el implementar la topología propuesta en esta tesis mediante distintas variantes de árboles de decisión podría permitir encontrar mejoras al modelo, por ejemplo, si al realizar las pruebas adecuadas, se determina que la construcción de una red MLP con la topología propuesta proporciona un porcentaje de clasificación aceptable tanto con un árbol de decisión podado como uno que no lo está, entonces utilizar el árbol podado reduciría aun más tanto el número de conexiones como el de neuronas ocultas en la red.

En problemas de clasificación, elegir qué clasificador es conveniente para un problema IOM específico depende principalmente de la naturaleza de los datos, se pueden diseñar distintos experimentos que permitan establecer condiciones que los datos deban cumplir para determinar cuándo es adecuado el uso de una red MLP con la topología propuesta. Por mencionar algunos ejemplos de experimentos con dicho objetivo, se propone el evaluar el desempeño de las *IG nets* construidas exclusivamente con conjuntos de datos de gran tamaño ó considerando conjuntos de datos que, independientemente del tamaño, generen un árbol de decisión de grande.

APÉNDICE A

LA CLASE "*CLASSIFIERTREE*"

La clase *ClassifierTree.java* de la librería Weka permite instanciar árboles de decisión con los atributos necesarios y operaciones para su uso, creados bajo el algoritmo C4.5. La estructura de un árbol instanciado con ésta clase está definida recursivamente, es decir, la raíz del árbol va apuntar a sus hijos que son subárboles, los cuáles también apuntaran a sus hijos y así sucesivamente hasta llegar a subárboles que no tengan hijos. En la figura 16 se muestran los atributos y operaciones más importantes para ésta clase, marcando aquellos que fueron agregados para hacer posible la implementación de los modelos *IG net* y *Entropy net.* Un árbol con al menos un hijo tiene el atributo *m_isLeaf* con valor *false*, en caso contrario su valor es *true*. La operación *assignIDs*, como su nombre lo indica, asigna un identificador a cada sub-árbol, haciendo un recorrido en preorden empezando de la raíz, esta asignación permite, por ejemplo, realizar la operación *graph* que proporciona una descripción de tipo *String* del árbol de decisión.

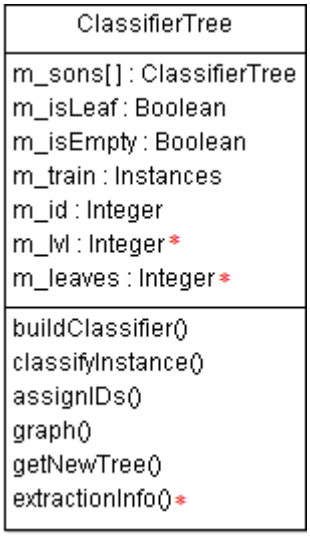

Figura 16. Principales atributos y operaciones de la clase *ClassifierTree.java.* El símbolo '*' indica atributos y operaciones agregadas para la implementación del modelo *IG net.*

Los objetos del conjunto de entrenamiento que recaen en un nodo del árbol están definidos en el atributo *m_train,* en caso de que no cumplan las condiciones para definir el nodo como hoja (ver apartado 2.2.2), en el método *buildClassifier* se determina qué atributo será asignado al nodo y el número de hijos que tendrá, donde cada nodo nuevo del árbol es creado con la operación *getNewTree*.

Los atributos y operaciones que fueron agregados permiten la implementación de redes MLP con modelo *IG net,* la justificación de los atributos es explicada en el apartado 2.2.2. y la idea para la implementación de la operación *extractionInfo* en el apartado 2.2.3.

APÉNDICE B

DESCRIPCIÓN DE ESCENARIOS

A.1. REQUERIMIENTOS DE USUARIO

El sistema *PCN – Tree (Partial Connected Nets – Tree*) permitirá construir y evaluar el desempeño de evaluación de tres posibles modelos de red MLP; *Entropy nets*, *IG nets* y la estándar, entrenadas con el algoritmo *backpropagation.*

A.2. REQUERIMIENTOS ESPECÍFICOS

- **Requerimientos Funcionales.**

RF-1. Elegir datos de entrenamiento.

RF-2. Elegir datos de prueba.

RF-3. Seleccionar Modelo Neuronal.

RF-4. Modificar parámetros de entrenamiento.

RF-5. Entrenar y evaluar.

RF-6. Visualizar desempeño de red.

- **Requerimientos No Funcionales.**

RNF-1. Acerca de. El usuario podrá obtener información general sobre el programa.

A.3. DIAGRAMA DE CASOS DE USO

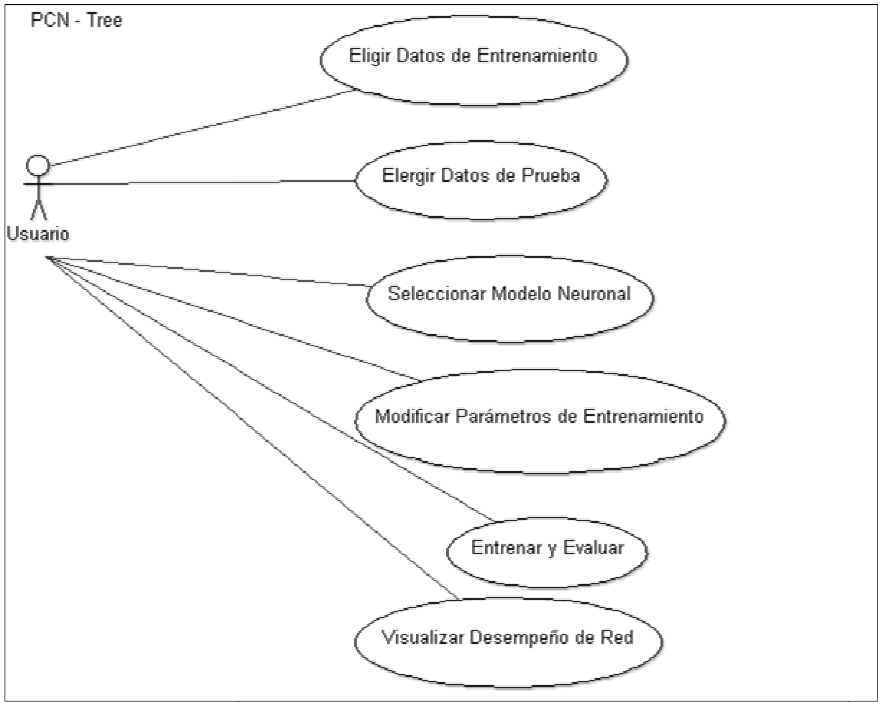

Figura 17. Diagrama de casos de uso del sistema PCN-Tree.

A.4. DESCRIPCIÓN DE ESCENARIOS DE CASOS DE USO

NOMBRE	**Elegir Datos de Entrenamiento**	
RF – 1	1	
Versión	1.0 24 abril 2012	
Autores	Dalia Rodríguez	
Objetivos asociados	Definir datos con los que se entrenará la(s) red(es).	
Descripción	El usuario especifica los datos con los que se entrenará la(s) red(es).	
Precondición	Ninguna	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario, da clic en el botón "Abrir…" en el panel "1. Datos de entrenamiento" desplegándose la ventana para seleccionar el archivo que contiene los datos de entrenamiento.
	2	El usuario, selecciona el archivo.
Postcondición	Los datos de entrenamiento están disponibles para su uso.	
Excepciones	**Paso**	**Acción**
	1	Si los datos de entrada no tienen el formato definido por WEKA, se mostrará un mensaje de aviso para el usuario indicando lo siguiente: "el formato de los datos es incorrecto, verifique el formato e intente nuevamente".

NOMBRE	**Elegir Datos de Prueba**
RF – 2	2
Versión	1.0 24 abril 2012
Autores	Dalia Rodríguez
Objetivos asociados	Definir datos con los que se entrenará la(s) red(es).
Descripción	El usuario especifica los datos con los que se evaluará el desempeño de la(s) red(es).
Precondición	Ninguna.

Secuencia Normal	**Paso**	**Acción**
	1	El usuario, deberá seleccionar entre una de tres posibles opciones ubicadas en el panel "2. Opciones de Prueba" las cuales pueden ser: "Usar el Conjunto de Entrenamiento", "Realizar Validación Cruzada", "Especificar un Archivo".
	1.1	Si el usuario selecciona "Especificar un Archivo" entonces deberá dar clic en el botón "Abrir Archivo" en el panel "2. Opciones de Prueba" desplegándose la ventana para seleccionar el archivo que contiene los datos de prueba y selecciona el archivo.
	1.2	Si el usuario selecciona "Validación Crazada" tendrá la opción de modificar el número de folds.
Postcondición	Los datos de prueba están disponibles para su uso.	
Excepciones	**Paso**	**Acción**
	1	Si el usuario seleccionó "Especificar Archivo" y el archivo no tiene el formato definido por WEKA, se mostrará un mensaje de aviso para el usuario indicando lo siguiente: "el formato de los datos es incorrecto, verifique el formato e intente nuevamente".
Comentarios	El default es "Usar el Conjunto de Entrenamiento".	

NOMBRE	**Seleccionar Modelo Neuronal**	
RF – 3	3	
Versión	1.0 24 abril 2012	
Autores	Dalia Rodríguez	
Objetivos asociados	Definir modelo neuronal para entrenar y evaluar.	
Descripción	El usuario selecciona el modelo de red del cual desea conocer su desem-peño de clasificación.	
Precondición	Ninguna.	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario, deberá seleccionar entre una de tres posibles opciones ubicadas en el panel "3. Tipo de Modelo" las cuales pueden ser: "IG net", "Entropy net" ó "MLP".

Postcondición	La red que se entrenará y evaluará estará definida, mostrando al usuario una breve reseña de dicha topología.
Excepciones	Ninguna

NOMBRE	**Modificar Parámetros de Entrenamiento**	
RF – 4	4	
Versión	1.0 24 abril 2012	
Autores	Dalia Rodríguez	
Objetivos asociados	Modificar variables libres del algoritmo *backpropagation* con el que se entrenará la(s) red(es).	
Descripción	El usuario modifica los parámetros del algoritmo de entrenamiento con los que se entrenará la(s) red(s).	
Precondición	Ninguna	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario hará clic en el botón "Configurar…" para desplegar la ventana donde podrá modificar tres variables libres "tasa de aprendizaje", "épocas" y "momentum".
	2	El usuario modifica los parámetros deseados.
	3	El usario da clic en "aceptar".
Postcondición	Ninguna	
Excepciones	Ninguna	
Comentarios	El default de los parámetros es 0.3 para "tasa de aprendizaje", 1000 para "épocas" y 0.2 para "momentum".	

NOMBRE	**Entrenar y Evaluar**
RF – 5	5
Versión	1.0 24 abril 2012

Autores	Dalia Rodríguez	
Objetivos asociados	Iniciar entrenamiento y evaluar el aprendizaje.	
Descripción	El usuario inicia el proceso de aprendizaje y evaluación de la red con la topología seleccionada.	
Precondición	RF-1, RF-2, RF-3	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario selecciona en el botón "Ejecutar Experimento" del panel "Tipo de Modelo".
Postcondición	Se muestra al usuario información referente al desempeño de la red entrenada y evaluada.	
Excepciones	**Paso**	**Acción**
	1	Si el usuario da clic en “Ejecutar Experimento" y la topología de red no fue seleccionada con anterioridad se le mostrará al usuario un mensaje de error: " Tipo de red no seleccionada"

NOMBRE	**Visualizar Desempeño de Red**	
RF – 6	6	
Versión	1.0 24 abril 2012	
Autores	Dalia Rodríguez	
Objetivos asociados	El usuario podrá consultar información del desempeño de la(s) rede(s) que ha evaluado y entrenado cuando así lo desee.	
Descripción	El usuario selecciona pestañas con la información del desempeño de clasificación.	
Precondición	RF-1, RF-2, RF-3, RF-5	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario, selecciona entre 4 posibles pestañas; "IG Net", "Entropy Net", "MLP" y "Resumen".
Postcondición	Ninguna	
Excepciones	Ninguna	

En la figura 18 se presenta la interfaz gráfica del sistema PCN-Tree, señalando los objetos que permiten la ejecución de cada requerimiento.

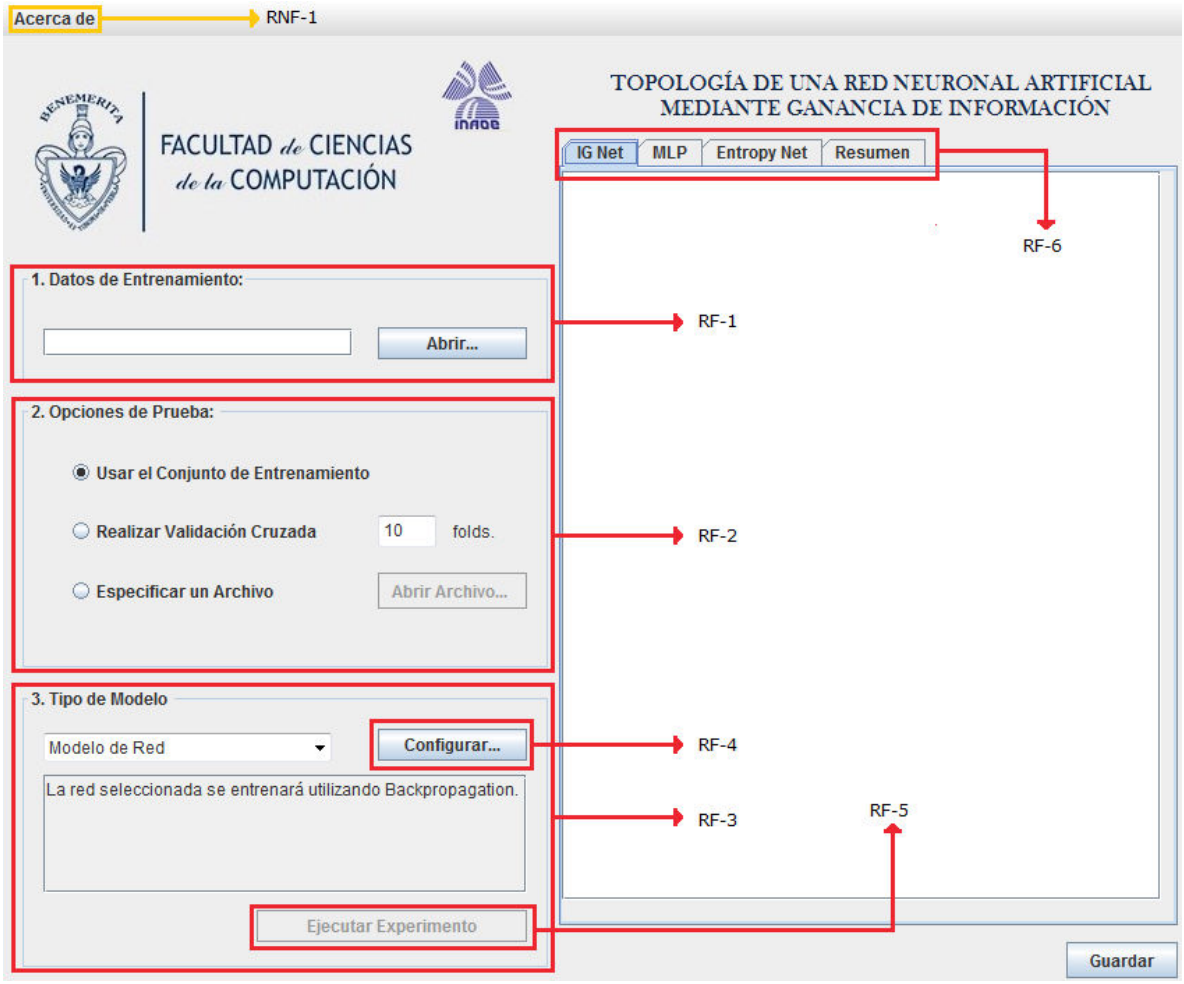

Figura 18. Interfaz gráfica del sistema PCN-Tree, indicando objetos que permiten ejecutar los requerimientos.

APÉNDICE C

VERIFICACIÓN DE LA IMPLEMENTACIÓN DE MODELOS

El sistema PCN-Tree implementa tres modelos de red MLP; *IG Net*, *Entropy Net* y el modelo con la topología estándar. Este último es construido sin realizar ninguna modificación a la librería Weka que lo implementa en la clase *MultilayerPerceptron.java,* por esta razón sólo se verificarán los modelos *IG Net* y *Entropy Net*. Para verificar que la implementación de estos dos modelos es correcta se ejecutaron dos experimentos utilizando el conjunto de datos *Wine* obtenido de *UCI Machine Learning Repository* [2010, Frank et al.]. Este conjunto de datos consta de 178 objetos descritos por 13 atributos denotados con las letras de la a a la m y los objetos pueden pertenecer a tres posibles clases denotadas como 1, 2 y 3.

B.1. MODELO IG NET

El árbol de decisión construido con el algoritmo C4.5 que la librería Weka devuelve para el conjunto de datos *Wine* es mostrado en la figura 19(a) y su representación gráfica se muestra en la figura 19(b), donde el valor de las ramas no fue representado por motivos de espacio, además, los valores de las ramas no aportan información para la construcción de la red con modelo *IG net*. El árbol decisión no fue podado y se utilizó un filtro para discretizar los atributos, implementado en la clase *Discretize.java* de la librería Weka previamente a la construcción del árbol.

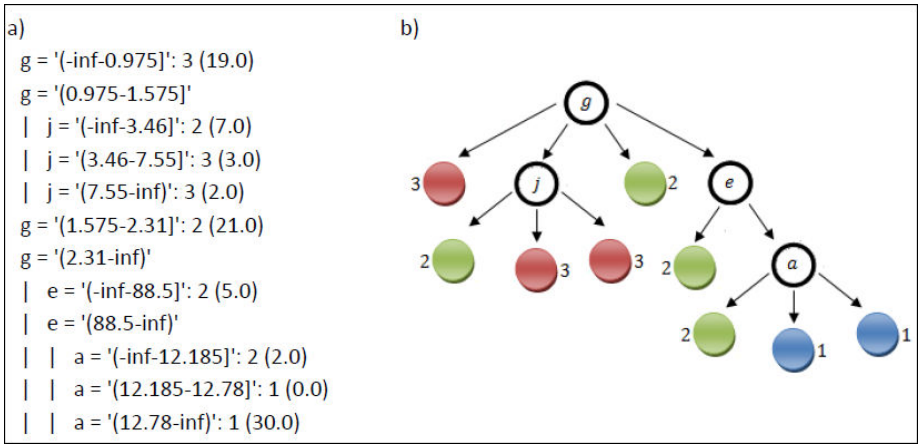

Figura 19. a) Descripción del árbol de decisión construido por la librería Weka para el modelo IG Net. **b)** Representación gráfica de a).

La salida del sistema PCN-Tree devuelve en la pestaña "*IG net"* la descripción de la red neuronal construida, dicha descripción es mostrada en la figura 20(a), su representación gráfica es mostrada en la figura 20(b). Al aplicar las reglas descritas en el capítulo 3.1. al árbol de decisión de la figura 20(b), se obtiene la misma red MLP parcialmente conectada mostrada en la figura 21(b) que corresponde al modelo de una *IG Net*.

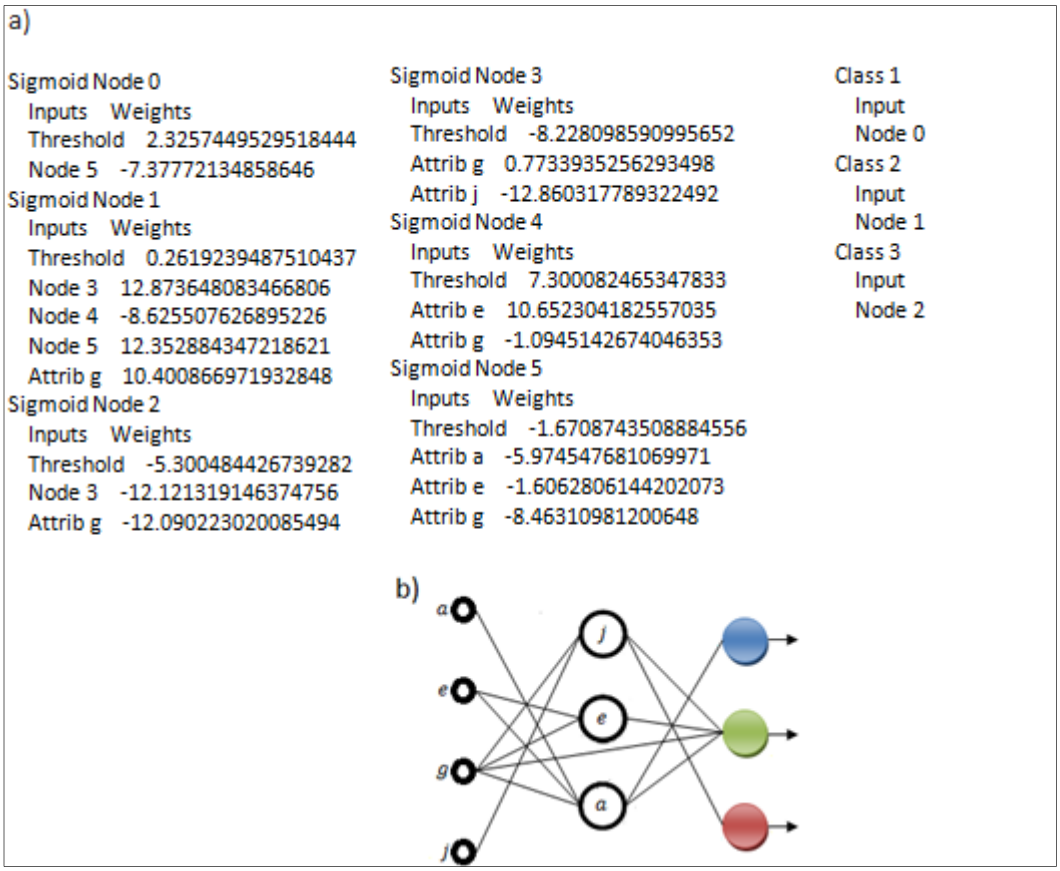

Figura 20. a) Descripción de la red construida por el sistema PCN-Tree con modelo IG Net. **b)** Representación gráfica para a).

B.1. MODELO ENTROPY NET

Utilizando el conjunto de datos *Wine* para construir un árbol de decisión binario y podado con el algoritmo C4.5 de la librería Weka se obtiene el árbol descrito en la figura 21(a) y su correspondiente representación gráfica en la figura 21(b). Al aplicar las reglas descritas en el capítulo 2.3.1 se obtiene la red MLP parcialmente conectada con el modelo de una Entropy Net que se muestra en la figura 22(b), el cual corresponde a la descripción de la red construida con el sistema PCN-Tree mostrada en la figura 22 (a).

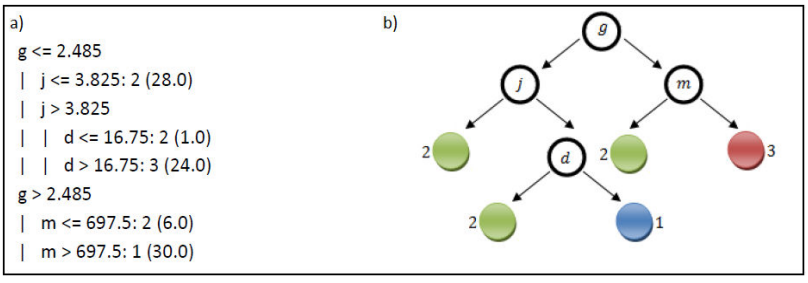

Figura 21. a) Descripción del árbol de decisión construido por la librería Weka para el modelo Entropy Net. **b)** Representación gráfica de a).

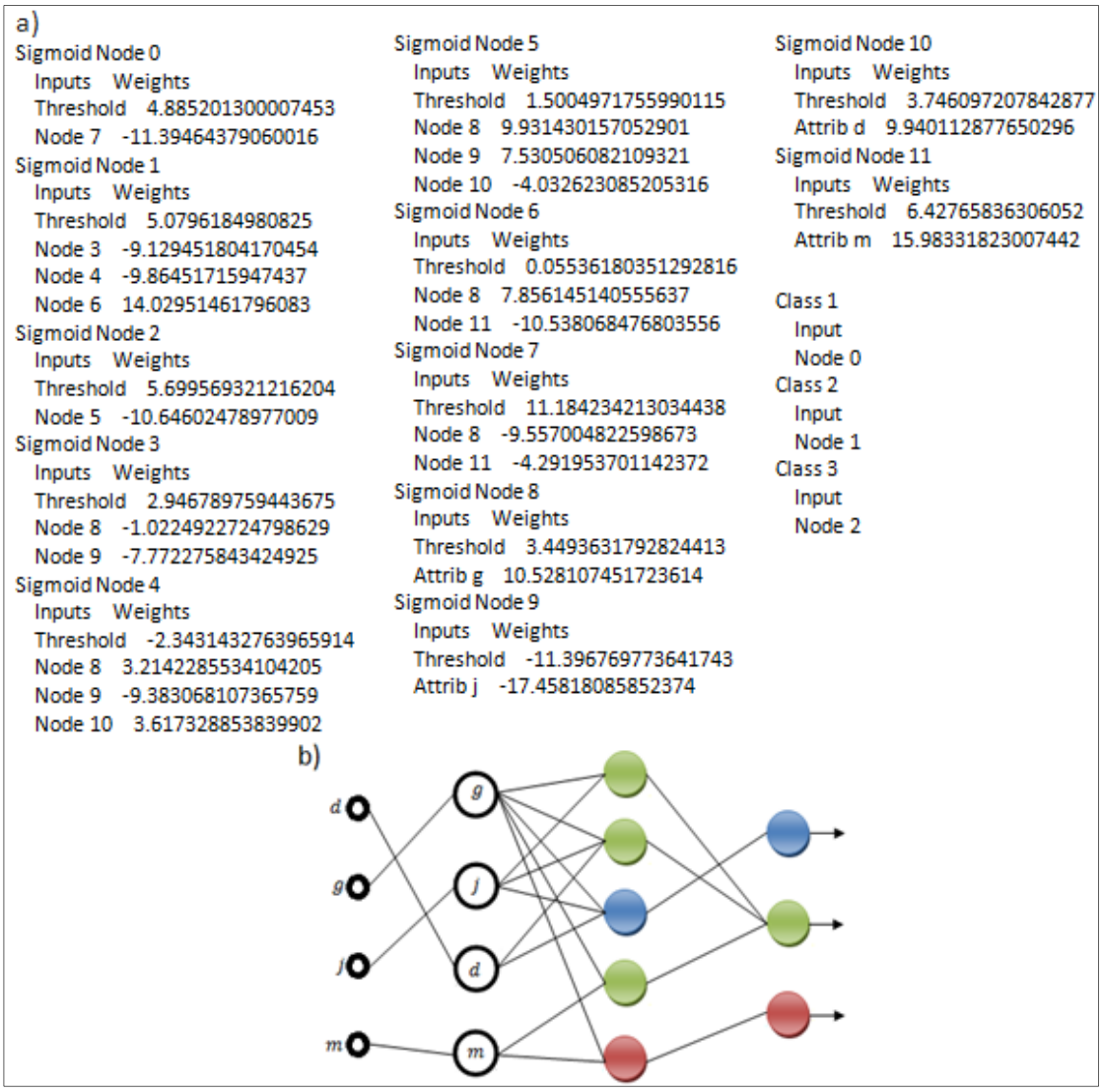

Figura 22. a) Descripción de la red construida por el sistema PCN-Tree con modelo Entropy Net. **b)** Representación gráfica para a).

APÉNDICE D

MANUAL DE USUARIO

Para construir y evaluar el desempeño de una red MLP con alguno de los tres modelos que permite el sistema PCN-Tree se deben seguir tres pasos, los cuales son descritos basándose en la interfaz mostrada en la figura 23. En la figura 23(a) se muestra la interfaz general del sistema y en la figura 23(b) la interfaz que permite modificar los parámetros de entrenamiento de la red.

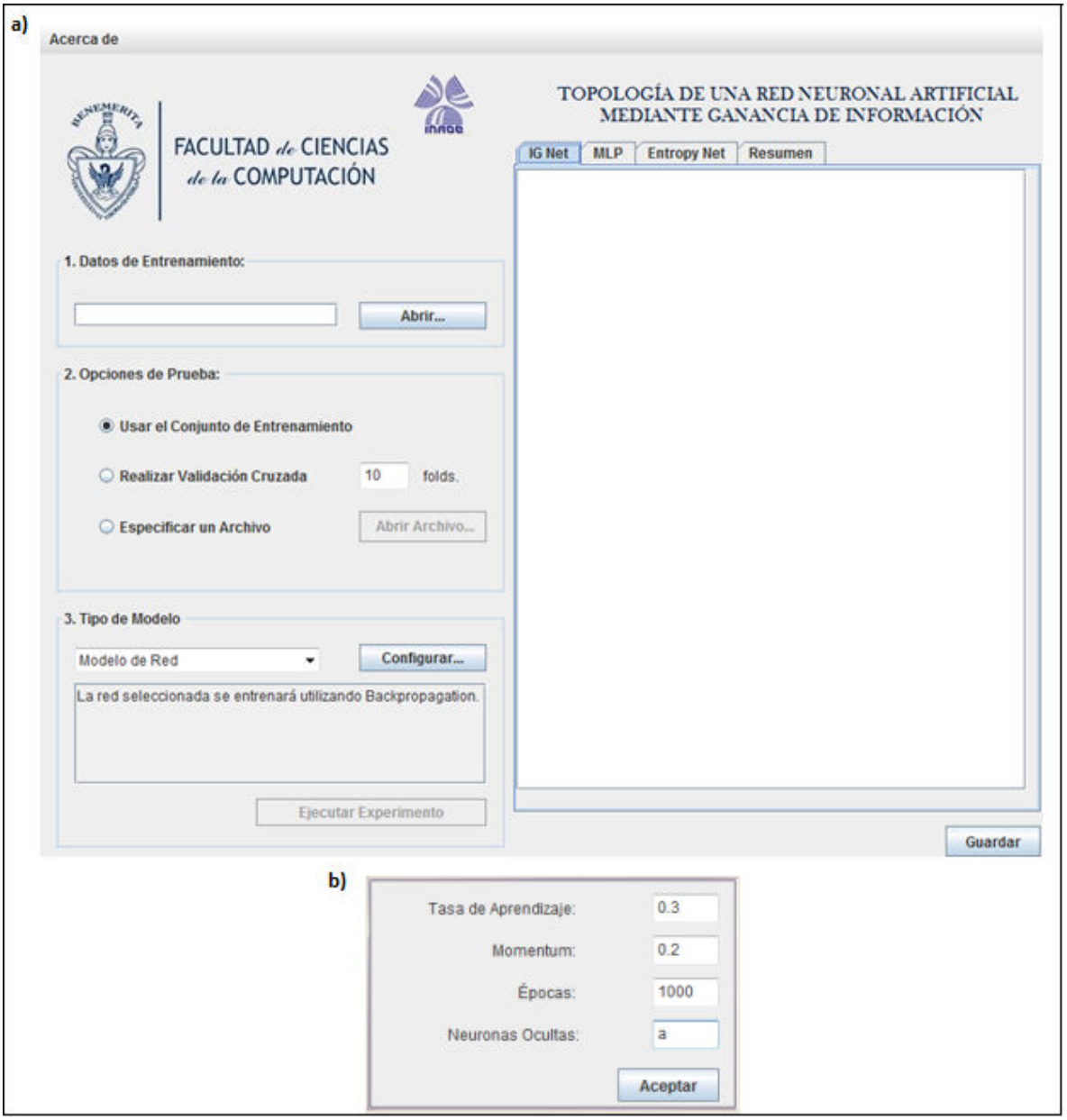

Figura 23. a) Interfaz gráfica del sistema PCN-Tree. **b)** Ventana de configuración de parámetros de entrenamiento.

Paso 1. En el panel **1. Datos de Entrenamiento** dar clic en el botón **Archivo…** a continuación seleccione el archivo que contiene los datos de entrenamiento con el formato de Weka que puede ser consultado en [2009, Hall et al.].

Paso 2. En el panel **2. Opciones de Prueba,** dar clic en la opción deseada para determinar los datos con los que se evaluará el desempeño de la red. Si selecciona la opción **Realizar Validación Cruzada** puede modificar el número de folds a realizar en el caja de texto ubicada junto a esta opción. En caso de seleccionar la opción **Especificar un Archivo** dar clic en el botón **Abrir Archivo…** a continuación seleccione el archivo con los datos de prueba con el formato de Weka.

Paso 3. En el panel **3. Tipo de Modelo** dar clic en la lista desplegable para seleccionar el modelo de red MLP que desea construir y evaluar. Puede modificar los parámetros de entrenamiento, si así lo desea dar clic en el botón **Configurar…** donde puede editar los valores deseados, a continuación dar clic en **Aceptar**. Si en el paso 2 seleccionó el modelo MLP, podrá modificar el número de capas y neuronas ocultas, escribiendo el número de neuronas que desea por capa empezando por la primera y separando con comas. Por ejemplo, si desea dos capas ocultas con 15 y 25 neuronas respectivamente escriba en el cuadro de texto correspondiente 15,25.

Una vez seleccionado el modelo y los parámetros de entrenamiento, dar clic en el botón **Ejecutar Experimento**. La descripción de la red construida se mostrará en la pestaña correspondiente al modelo seleccionado. En la pestaña **Resumen** se muestra para cada modelo construido un sumario de los datos de entrenamiento, del modelo construido y de la evaluación del modelo.

BIBLIOGRAFÍA.

- Breiman L., Friedman J.H., Olshen R.A., Sotne C.J., "Classification and regression trees", Wadsworth International Group, Belmount, CA, The Wadsworth Statistics/Probability Series, 1984.
- Cybenko G. “Approximation by superpositions of a sigmoidal function”, Technical Report, Urbana, IL, 1988.
- Demšar Janez, “Statistical Comparisons of Classifiers over Multiple Data Sets”, Faculty of Computer and Information Science, Slovenia, 2006.
- Dietterich Thomas G., “Approximate Statistical Test for Comparing Supervised Classification Learning Algorithms”, Oregon State University, 1997.
- Elizondo D., Fiesler E., "A survey of partially connected neural networks", Int. J. Neural Syst., vol. 8, no. 5 y 6, pp. 535-558, 1997.
- Fayyad Usama M., Irani Keki B., "Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning", proc. 13th International Joint Conference on Articial Intelligence, pp. 1022-1027, 1993.
- Frank A., Asunción A., “UCI Machine Learning Repository”, CA: Uni-versity of California, School of Information and Computer Science, http://archive.ics.uci.edu/ml, 2010.
- Gómez-Gil Pilar. "Notas del curso Tópicos Avanzados: Redes Neuronales Artificiales", INAOE, úlitma actualización: 17 de Enero de 2012, consultado: 8 de Febrero de 2012, disponible en: http://ccc.inaoep.mx/~pgomez.
- Hall Mark, Frank Eibe, Holmes Geoffrey, Pfahringer Bernhard, Peter Reutemann, Witten Ian H., “The WEKA Data Mining Software: An Update”, SIGKDD Explorations, vol. 11, no. 1, pp. 10-18, 2009.
- Haykin Simon, “Neural Networks and Learning Machines”, ed. Macmillan Collage Publishing Company, 3ª edición, McMaster University, Canadá, 2009.
- Holte Robert C., "Very Simple Classification Rules Perform Well on Most Commonly Used Datasets", Springer, vol. 11, no. 1, pp. 63-91, 1993.
- Kang Sanggil, Isik Can. “Partially Connected Feedforward Neural Networks Struc-tured by Input Types”, Trans. IEEE, vol. 16, no. 1, 2005.
- Konar Amit. “Artificial Intelligence and Soft Computing: Behavioral and Cognitive Modeling of the Human Brain”, editorial CRC. 1ª edición, 2000.
- Kotsiantis S. B.,"Decision trees: a recent overvier", Springer Netherlands, 2011.

- McCulloch, Pitts, “A logical calculus of the ideas immanent in nervous activity”, Bulletin of Mathematical Biophysics, 1943.
- Mehta M, Agrawal R, Riassnen J., "SLIQ: a fast scalable classifier for data mining" , Springer, vol. 1057, pp. 18-32, 1996.
- Mitchell Tom M., "Machine Learning", McGraw Hill, 1997.
- Pajares Martinsanz Gonzalo, Santos Peñas Matilde, “Inteligencia Artificial e Ingeniería del Conocimiento”, editorial Alfaomega, 1ª edición, 2005.
- Quinlan J. R., "C4.5: Programs for Machine Learning", Morgan Kaufmann Publishers , 1993.
- Rumelhart D. E., Hinton G. E., Williams R. J., “Learning internal representations by error propagation”, proc. Explorations in the Microstructure of Cognition, vol. 1, Massachusetts, 1986.
- Sethi Ishwar K., "Entropy Nets: From Decision Trees to Neural Networks", Proc. IEEE, vol. 78, no. 3, pp. 1605-1613, 1990.
- Shafer J., Agrawal R., Mehta M.,"SPRINT: a scalable parallel classifier for data mining",1996, In Proc. 22nd VLDB conference, Bombay, 1996.
- Sierra Araujo Basilio, “Aprendizaje Automático: conceptos básicos y avanzados”, editorial Pearson Educación, 1ª edición, 2006.
- Tulach, Jarda, “Netbeans Integrated Development Environment”, 2011, actualmente bajo licencia de Oracle.
- Werbos P., "Beyond Regression: New tools for prediction and Analysis in the Behavioral Sciences", Ph.D. thesis, Harvard University, 1974.