UNIVERSIDAD SIMÓN BOLÍVAR
Decanato de Estudios de Postgrado
Maestría en Ingeniería de Sistemas

TRABAJO DE GRADO

MÉTODOS MULTI-SUPERFICIE PARA CONSTRUIR CLASIFICADORES BINARIOS CON OPTIMIZACIÓN MATEMÁTICA

por

Ingº Orestes Gonzalo Manzanilla Salazar

Octubre, 2005

UNIVERSIDAD SIMÓN BOLÍVAR
Decanato de Estudios de Postgrado
Maestría en Ingeniería de Sistemas

MÉTODOS MULTI-SUPERFICIE PARA CONSTRUIR CLASIFICADORES BINARIOS CON OPTIMIZACIÓN MATEMÁTICA

Trabajo de Grado presentado a la Universidad Simón Bolívar por

Ingº Orestes Gonzalo Manzanilla Salazar

Como requisito parcial para optar al grado de

Magister en Ingeniería de Sistemas

Realizado con la tutoría del Profesor Ubaldo Manuel García Palomares

Octubre, 2005

UNIVERSIDAD SIMÓN BOLÍVAR

Decanato de Estudios de Postgrado

Maestría en Ingeniería de Sistemas

MÉTODOS MULTI-SUPERFICIE PARA CONSTRUIR CLASIFICADORES BINARIOS CON OPTIMIZACIÓN MATEMÁTICA

Este Trabajo de Grado ha sido aprobado en nombre de la Universidad Simón Bolívar por el siguiente jurado examinador:

Presidente

Prof. Miguel Strefezza

Miembro Externo

Prof. Marcos Raydan

Universidad Central de Venezuela

Miembro Principal-Tutor

Prof. Ubaldo García Palomares

Fecha: 27-10-05

Dedicatoria

Las matemáticas,

como cualquier cosa que toquen nuestras manos,

puede usarse en la búsqueda de calidad de vida,

o para perfeccionar la destrucción.

Dedico lo que de mi ser he plasmado aquí,

a quienes empeñan sus manos

en dar vidas,

no en quitarlas;

en alegrar rostros,

no en apagarlos,

y en elevar pensamientos,

en vez de envanecerlos.

Agradecimientos

Los aportes de este trabajo son respuesta a las motivadoras interrogantes que surgieron en las clases dictadas por mi tutor, el Prof. Ubaldo M. García Palomares, quien me ha guiado y formado con su ejemplo y sus ideas en el proceso de realizar esta investigación, trazando caminos sobre los terrenos sembrados por valiosos precursores del área de optimización.

En numerosas ocasiones durante ese proceso, fueron las palabras de aliento y orientación de la Prof. Borges, y mis compañeros del informalmente llamado “Grupo de Optimización” de la USB, Claudia e Ildemaro, las que me permitieron seguir adelante. La Prof. Isabel Llatas, además de la Prof. Borges y el Prof. Ubaldo, conforman el que he denominado mi “comité de padrinos” dentro del profesorado de la USB, ya que fue por sus consejos sabios y apoyo oportuno, que emprendí empresas ahora muy importantes para mi, como el haber realizado este postgrado, y haber sido ayudante académico de esta casa de estudios. Empresas que, por cierto, hubiesen sido difíciles sin el apoyo del Centro de Estadística y Software Matemático de la USB, al que, como miembro asociado, le debo mucho por su infraestructura y su ameno ambiente de intercambio de ideas. Entre sus miembros debo nombrar a Jhan Rodríguez y a Nora Monsalve pues las conversaciones con ellos arrojaron luces sobre muchos aspectos del trabajo. Los conocimientos básicos aprendidos sobre las redes neurales los obtuve en primera instancia gracias al curso de la Prof. Minaya Villasana y del Prof. Oscar Rondón, del Dpto. de Estadística y Cómputo Científico.

Las enseñanzas de vida que me proporcionó el Prof. Guillermo Álvarez me han permitido gestionar esta reciente etapa de mi vida, con una fluidez que nunca imaginé, tanto en lo académico, como en lo profesional, y lo personal. Mi adorada Juvili Hernández, fue mi compañera de infinitas noches frente al computador, fuente de entusiasmo y de invaluables interrogantes y reflexiones generadoras de aportes (algunos de ellos incorporados en la esencia de las propuestas de este Trabajo de

Grado), hechas desde una mente limpia de paradigmas. También fue mi ayudante en el diseño de mis ilustraciones, y en correcciones de estilo y me hizo crecer inmensamente en el aspecto personal, principalmente en lo que a mi auto-conocimiento se refiere. Antes de referirme a mis padres, quisiera decir que en muchos momentos estuve en grave riesgo de formar parte de los dispositivos “plug-and-play” de mi computador, de no ser por mis amigos, o hermanos a quien la vida no dio mi sangre. Con ello me refiero a mis entrañables “Angélicos”, a Virginia, a Luisana, a Félix, a Alberto, a mi gente de Pachi Taichi Chuan, y a tantos otros que lograron que siguiera formando parte de la humanidad como ser social sano. Ahora si, quisiera agradecer infinitamente a mis padres, Ludmila Salazar y Orestes Manzanilla Sáez, por sus igualmente infinitos dones de paciencia, amor, y solidaridad. Sin ellos no existiría, ni sería el ser que soy. Ellos me han permitido sentirme, hasta el día de hoy, novio de la vida, aunque ellos mismos hayan sospechado en algunos momentos que este Trabajo de Grado me alejaba de ellos.

Finalmente, manifiesto estar consciente de la “in-completitud” de los reconocimientos hechos en esta página, y por eso los extiendo a todas las personas no nombradas que han querido que mi Trabajo de Grado y yo formemos parte de sus vidas. He de agradecer y honrar a quienes hayan percibido el amor con que escribo estas líneas.

RESUMEN

El objetivo de este trabajo es proponer un tipo más general de algoritmo multi-superficie, reconociendo en un algoritmo particular precedente, el MSM de O.L. Mangasarian, una innecesaria “rigidez”. El propuesto es un método para construir y entrenar redes neurales artificiales de una capa oculta, destinadas a la clasificación binaria de conjuntos disjuntos de elementos, seccionando el espacio de clasificación con múltiples hiper-superficies. Se basa en el análisis de las restricciones “innecesarias” del MSM, cuya efectividad ha sido probada en casos reales. El MSM es considerado un caso particular del método general propuesto. Un subproducto del algoritmo es el número de neuronas de la capa oculta de la red, que corresponde al número de hiper-superficies utilizadas. Para evaluar el principio de “flexibilización” implícito en el algoritmo general, se genera una instancia del mismo, para comparar su desempeño con el del MSM. Se consideran sólo casos donde las superficies son linealmente dependientes de los parámetros que encuentra el algoritmo. Los mismos se obtienen mediante optimización no relacionada a gradientes. Pruebas numéricas en casos sintéticos, mostraron una mayor efectividad de clasificación con una menor cantidad de neuronas por parte de la instancia generada. Se discuten criterios para minimización de errores, maximización de la capacidad de generalización del clasificador binario, y estrategias para evadir soluciones triviales y degeneradas. Se abre camino para la creación de diversas instancias del método general propuesto en investigaciones posteriores, más eficientes, versátiles y variadas, que los algoritmos multi-superficie existentes a la fecha. La propuesta aboga a favor de los clasificadores lineales por partes, frente a los clasificadores no-lineales como las Máquinas de Vectores de Soporte (SVM).

Palabras Clave: Clasificación Binaria, Entrenamiento Supervisado, Programación Matemática, Maquinas de Vectores de Soporte, Redes Neurales Artificiales.

Índice General

1. INTRODUCCIÓN 1

1.1 Antecedentes del problema 5

1.2 Planteamiento y justificación del problema 14

1.3 Sobre la organización del trabajo 17

2. MARCO TEÓRICO 19

2.1 Redes Neuronales Artificiales 19

2.1.1 Definición 19

2.1.2 Relevancia Práctica 20

2.1.3 Usos de las RNAs 22

2.1.3 RNAs de realimentación positiva 24

2.1.4 Neuronas tipo Perceptrón y Perceptrones Multi-Capa 24

2.1.5 El “escalón” como Función de Transferencia 25

2.1.6 Arquitectura de los Perceptrones Multi-Capa 27

2.1.7 Aprendizaje de las Redes Neurales Artificiales 29

2.2 Perceptrones como clasificadores binarios lineales 33

2.2.1 Conjuntos Linealmente Separables e Inseparables 33

2.2.2. Construcción de Perceptrones Binarios Multi-Capa 34

2.2.3 Método Multi-Superficies de Mangasarian 39

3. MÉTODOS Y PROCEDIMIENTOS UTILIZADOS 47

3.1 Planteamiento del Problema 47

3.2 Planteamiento de la Propuesta del Método General 49

3.3 Nomenclatura de los Métodos Multi-superficie 53

3.4 Instancia del Método General Propuesto 54

3.4.1 Comparación con el MSM 56

3.4.2 Modelo de Optimización 58

3.4.3 Comportamiento de las Superficies Generadas 59

3.4.4 Algoritmo OGMS-LGP 61

3.4.5 RNA Generada 63

3.4.6 Casos de Hiper-Planos Degenerados 66
3.4.7 Algoritmo de Eliminación de Hiper-Planos Degenerados 67
3.5 Sobre los criterios de optimalidad 71
3.6 Convergencia del OGMS-LGP 76
4. EXPERIMENTOS, RESULTADOS Y CONCLUSIONES 80
4.1 Selección del Algoritmo de Referencia 80
4.2 Selección del Criterio de Comparación 83
4.3 Diseño de los Experimentos 85
4.4 Resultados Computacionales 89
4.5 Conclusiones y Recomendaciones 92
4.5.1 Aportes Adicionales 93
4.5.2 Planes de Investigación 95
4.5.3 Recomendaciones 96
5. REFERENCIAS BIBLIOGRÁFICAS 99

Índice de Figuras

Figura 1: Riesgo Empírico vs. Riesgo de *Over-fitting* 4
Figura 2: RNA *Feed-forward* de una sola capa oculta. 6
Figura 3: Perceptrón o *Lineal Threshold Unit* 7
Figura 4: Hiper-plano separador generado por una LTU 8
Figura 5: Separación Robusta mediante Programación Lineal 10
Figura 6: Ejemplo de 2 Conjuntos disjuntos de elementos en R^2 12
Figura 7: Iteraciones 1ra, 2da, 3ra y 4ta del MSM 13
Figura 8: Iteración 5ta del MSM 14
Figura 9: Clasificación Binaria con dos Hiper-planos no paralelos 16
Figura 10: Superficie "por partes" generada por el algoritmo propuesto 17
Figura 11: Tres iteraciones del MSM 40
Figura 12: Árbol de Binario correspondiente al método MSM 41
Figura 13: RNA generada por el MSM 42
Figura 14: Ejemplo de dos conjuntos disjuntos 51
Figura 15: Hiper-superficie contentiva de los puntos del conjunto A 52
Figura 16: Hiper-superficie contentiva de los puntos del conjunto B 52
Figura 17: Par de Hiper-superficies 53
Figura 18: Puntos sin clasificar por Hiper-superficies 53
Figura 19: Tres iteraciones de la instancia propuesta 60
Figura 20: Árbol binario de la instancia generada 61
Figura 21: RNA generada por el OGMS-LGP 64
Figura 22: Ejemplo de iteración con un hiper-plano degenerado 67
Figura 23: Eliminación de hiper-plano degenerado sin cambiar función de transferencia 69
Figura 24: Cambio en función de transferencia para eliminar hiper-plano degenerado 70
Figura 25: Conjunto de datos "Avión" 87
Figura 26: Conjunto de datos "4-Cuñas" 87
Figura 27: Típica 1ra iteración del MSM en el conjunto de datos "Avión" 90

Figura 28: Típica 1ra iteración del OGMS-LGP en el conjunto de datos “Avión” 91
Figura 29: Típica 1ra iteración del MSM en el conjunto de datos “4-Cuñas” 91
Figura 30: Típica 1ra iteración del OGMS-LGP en el conjunto de datos “4-Cuñas” 92

Índice de Modelos

Modelo 1: PL para separación robusta de Mangasarian y Bennett 9
Modelo 2: Minimización del riesgo empírico de clasificación binaria 29
Modelo 3: PL que minimiza los elementos de $\mathcal{B}$ mal clasificados 55
Modelo 4: PL que minimiza los elementos de $\mathcal{A}$ mal clasificados 55
Modelo 5: PL ampliado para obtener 2 hiper-planos no-paralelos 56
Modelo 6: PL ampliado con restricción de paralelismo 57
Modelo 7: PL ampliado con indicador de signo 59
Modelo 8: PL ampliado con indicador de signo factorizado 59
Modelo 9: PL ampliado con minimización indirecta de pesos 76

Índice de Algoritmos

Algoritmo 1: MSM de Mangasarian.....44
Algoritmo 2: Procedimiento de Anti-degeneración del MSM 46
Algoritmo 3: Método General Propuesto 50
Algoritmo 4: OGMS-LGP 62
Algoritmo 5: Eliminación de hiper-planos Degenerados en OGMS-LGP 71
Algoritmo 6: Modificación al procedimiento de definición de descendientes de $\mathcal{A}$ y $\mathcal{B}$.....88

Índice de Ecuaciones

Ecuación 1: RNA denotada como grafo 19
Ecuación 2: Campo proyectivo de una RNA 20
Ecuación 3: Campo receptivo de una RNA 20
Ecuación 4: Función de transferencia de una LTU 25
Ecuación 5: Sigmoide clásica como función de transferencia 26
Ecuación 6: Tangente hiperbólica como función de transferencia 26
Ecuación 7: Equivalencia entre una sigmoide clásica y la tangente hiperbólica 27
Ecuación 8: Salida de un perceptrón 30
Ecuación 9: Convención de la respuesta de las LTUs del MSM 41
Ecuación 10: Conclusión de la prueba de convergencia del OGMS-LGP 78

Lista de Símbolos y Abreviaturas

$\alpha, \ldots, \omega$	Números reales,
$\mathcal{A}, \ldots, \mathcal{Z}$	Conjuntos,
$A, \ldots, Z$	Matrices,
A^k	Fila k de la matriz A,
$a, \ldots, h, u, \ldots, x, y, z$	Vectores,
e	Vector con componentes iguales a uno (1), de dimensión adecuada al contexto,
x^k	Componente k del vector x,
x_i	Vector x de la iteración i,
$i, \ldots, t$	Índices o números enteros,
$\langle\, , \rangle$	Producto Interno,
$\lvert \cdot \rvert$	Norma inducida por $\langle\, , \rangle$,
R^n	Espacio Euclidiano n dimensional,
$\{x_i\}$	Sucesión,
$<, \leq, =, \geq, >$	Denota símbolos de igualdad o desigualdad según la restricción involucrada.

1. INTRODUCCIÓN

"Plurality should not be assumed
without necessity"
- William de Ockham, siglo XIV.

El principio de la navaja de Ockham es una invitación
a reducir la complejidad de las explicaciones
que se dan a los fenómenos de la realidad
(o al menos no aumentarla
más allá de lo necesario).

El presente trabajo es una mejora a un algoritmo que ha sido propuesto para resolver problemas de clasificación binaria mediante redes neurales *feed-forward* (o de re-alimentación positiva) con una sola capa escondida de neuronas. Las mejoras presentadas se han ideado con la intención exhibir un algoritmo que sea a la vez competitivo y de fácil implementación, procurando que se distinga por generar la menor cantidad de entidades clasificadoras.

El problema de la clasificación es el de encontrar, basándose en "experiencias *a priori*", una regla general que sirva para asignar acertadamente una clase a "nuevas experiencias". El caso de la clasificación binaria es el más sencillo que puede plantearse, en el que sólo hay dos clases distintas.

Una manera sencilla, pero formal, de expresar el problema, es que se desea estimar una función $f : R^n \rightarrow \{-1,+1\}$, utilizando m pares conocidos de datos *entrada-salida*. Las "experiencias previas" están representadas por los pares de datos (x_i, c_i), con

$x_i \in R^n$, $c_i \in \{-1,+1\}$, $i = 1, \ldots, m$. La entrada $x_i \in R^n$ está compuesta por los atributos de ese elemento (donde n es el número de atributos), y la segunda componente, c_i, corresponde a la clasificación conocida *a priori.* En esta formalización del problema la clasificación está codificada utilizando la salida $c_i = -1$ para indicar la pertenencia de un elemento a uno de los conjuntos, y la salida $c_i = 1$ para indicar la pertenencia al otro conjunto. Se desea que la función, basándose en esos pares de datos, "aprenda". Ese "aprendizaje" permitiría que esa función pueda ser utilizada para clasificar nuevos elementos, con una efectividad razonable.

Pese a la sencillez de este tipo de problemas, sus aplicaciones han sido exitosas en diversos campos (como se menciona en [MMRTS01] y [BM92b]), entre ellos están:

- Reconocimiento de patrones ópticos
- Reconocimiento de patrones de voz
- Categorización de textos
- Análisis de perfiles de expresión genética
- Análisis de ADN y proteínas diversas
- Control robótico
- Problemas financieros
- Problemas de exploración petrolera
- Diagnóstico y prognosis médica.

Funciones como la que se desea estimar, pueden ser definidas, literalmente, de infinitas formas. Sin embargo, el criterio que generalmente se acepta (y que se adopta en este trabajo), es disminuir el riesgo de hacer una mala clasificación en una "experiencia *a posteriori*".

Este riesgo no puede ser minimizado directamente, porque habría que conocer todos los posibles entes a clasificar [MMRTS01] o determinar la función de densidad probabilística [Vap98]. Sin embargo, los intentos son dirigidos hacia la estimación de una función que cumpla *lo mejor que pueda* con ese propósito, basándonos en la

información disponible. A los entes de clasificación conocida a priori, utilizados para estimar esta función, se les denomina "conjunto de entrenamiento". Una forma de utilizar estos entes conocidos, es buscar con ellos el mínimo del riesgo empírico. Ello consiste en encontrar una función que clasifique estos objetos, con la menor cantidad de objetos mal clasificados. El mínimo del riesgo empírico tiende al mínimo del riesgo real, a medida que se aumenta el tamaño del conjunto de entrenamiento (tendiendo al infinito).

La infinita cantidad de funciones que pueden utilizarse como "máquina de aprendizaje", en general muestra que, *a mayor complejidad* de la función, *menor es el riesgo empírico*. Sin embargo, para pequeños conjuntos de aprendizaje, puede haber grandes desviaciones al aplicarse sobre nuevas entidades, ya que podría haberse provocado un *over-fitting* (sobre-ajuste o exceso de ajuste), lo cual se refiere a que la función pierde generalidad y resulta solamente aplicable para un conjunto dado de datos [MMRTS01].

Una forma de evadir el dilema del *over-fitting*, es restringir la complejidad de la función de clasificación. Resulta razonable pensar que una regla "simple" que explique la mayor parte de los datos es preferible a una compleja, y es en este punto donde se hace honor a la Navaja de Ockham. Típicamente se introduce un término de "regulación" que limita la complejidad de la función de clasificación [MMRTS01]. Sin embargo, el problema de saber cual es el término óptimo de regulación, resulta extremadamente difícil. La mejor estrategia yace en un compromiso entre ambos tipos de riesgo, de tal forma que ninguno de ellos sea muy grande. La situación ideal podría visualizarse como el punto donde interceptan las curvas de Riesgo Empírico y Riesgo de Over-fittig en la Figura 1, punto que corresponde al mínimo del riesgo "total" compuesto por ambos tipos de riesgo. Lamentablemente la curva del riesgo de *over-fitting* no puede ser conocida, sino que su construcción es puramente hipotética.

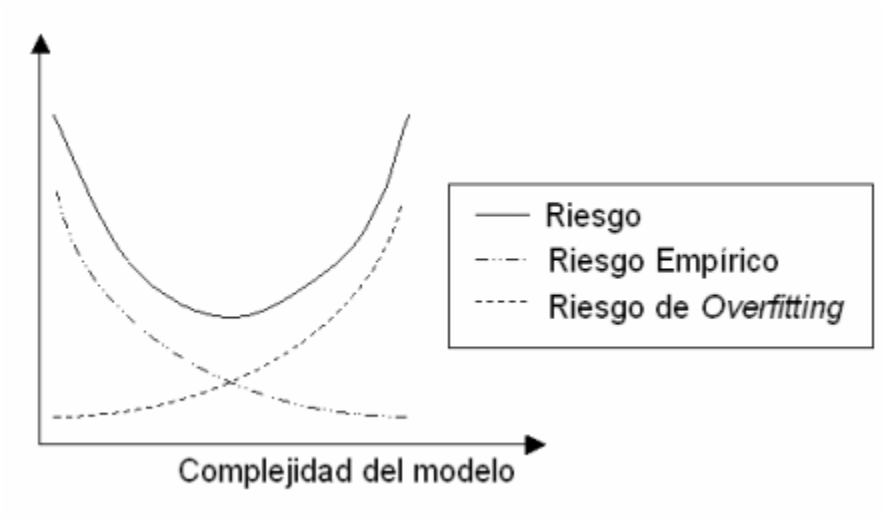

Figura 1: Riesgo Empírico vs. Riesgo de *Over-fitting*

En el caso del algoritmo propuesto en este trabajo, se asume la posición a favor del uso de las estructuras simples de separación que existen, como lo son los hiper-planos. Sin embargo, la extrema sencillez de estas estructuras hace que sean de poca utilidad para problemas de cierta complejidad, ameritando el uso de más de un hiper-plano o de funciones más complejas.

En la elaboración del algoritmo propuesto, se tuvo en mente la premisa de intentar el mejor acercamiento posible al número mínimo u óptimo, de hiper-planos, para resolver un problema dado. Es un hecho bien conocido que existe un número de hiper-planos que puede disminuir el error empírico hasta 0 (cero). Sin embargo, acercarse a ese número de hiper-planos podría aumentar el riesgo de incurrir en *over-fitting* (asociado a la falta de generalidad) [MMRTS01].

1.1 Antecedentes del problema

El problema de la clasificación binaria ha sido resuelto desde diferentes ámbitos, tales como:

- los métodos estadísticos de reconocimiento de patrones
- la Programación matemática (los *Support Vector Machines* son un caso típico), y
- la Inteligencia Artificial.

En el ámbito de la Inteligencia Artificial, el tema de las Redes Neurales Artificiales ha venido dotándose de gran riqueza y complejidad. De hecho, es común ver que se le denomine como un área de estudio distinto al de la Inteligencia Artificial, tomando un lugar propio como conjunto de estrategias de automatización de decisiones.

Desde el ámbito de las Redes Neurales Artificiales (en adelante RNA), el problema de la clasificación binaria ha sido descrito recientemente de una manera absolutamente equivalente a cierto tipo de modelos de programación matemática, en particular la Programación Lineal (en adelante PL) y la Programación Cuadrática (en adelante PC).

De hecho, el uso de PL para problemas de clasificación fue uno de los tópicos importantes del curso de Análisis de Sistemas Lineales, ofertado en el trimestre Enero-Marzo de 2004 en la USB para este programa de Maestría. De la dinámica profesor-alumnos que hubo en ese curso surgió la idea primigenia que hay detrás del planteamiento realizado en este trabajo de grado.

Una de las ventajas de este tipo de RNAs, que tienen equivalencia con modelos de PL, es que ha sido demostrado, tal y como lo señala [Man92], que las RNAs, y en particular las de una sola capa oculta (ver Figura 2) son capaces de separar

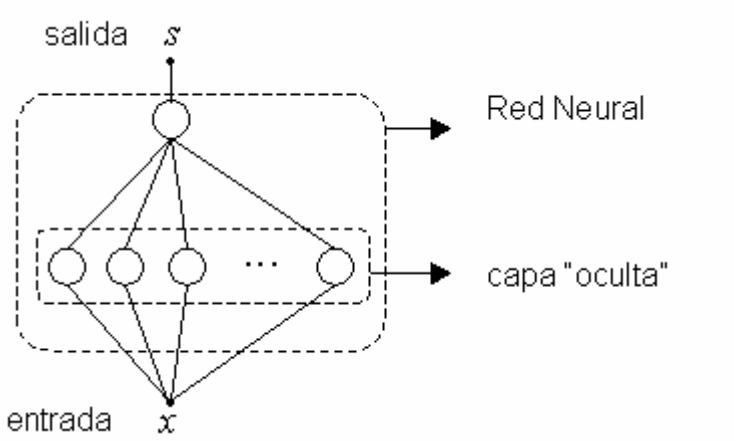

Figura 2: RNA *Feed-forward* de una sola capa oculta.

completamente cualesquiera par de conjuntos de puntos disjuntos en R^n, siempre y cuando tengan una cantidad suficiente de neuronas.

Para la construcción de este tipo de RNAs, se utilizan neuronas artificiales de un tipo bastante clásico, denominadas “perceptrones” o *Lineal Threshold Units* (Unidades de Umbral Lineal), que en adelante serán denominadas LTU. Ellas funcionan realizando una suma ponderada de las entradas, y arrojando como resultado un valor de s=+1 cuando dicha suma sobrepasa el valor del umbral, o un valor de s=0 en caso contrario. En algunas aplicaciones se prefiere el valor de s=–1 en lugar del s=0 para cuando el umbral no es sobrepasado. En el contexto de esta investigación ésa será la convención a utilizar. En la Figura 3 se encuentra una representación de un perceptrón, con sus distintos parámetros, así como dos formas alternativas de modelar la función umbral.

Las LTU pueden ser vistas como una representación de un hiper-plano que delimita dos semi-espacios de clasificación. Previamente a este trabajo ha habido resultados prácticos para encontrar los parámetros de estas LTU, como se ve en [MSW90], [WM90], y [BM92b]. En la Figura 4 se muestra un hiper-plano correspondiente a una

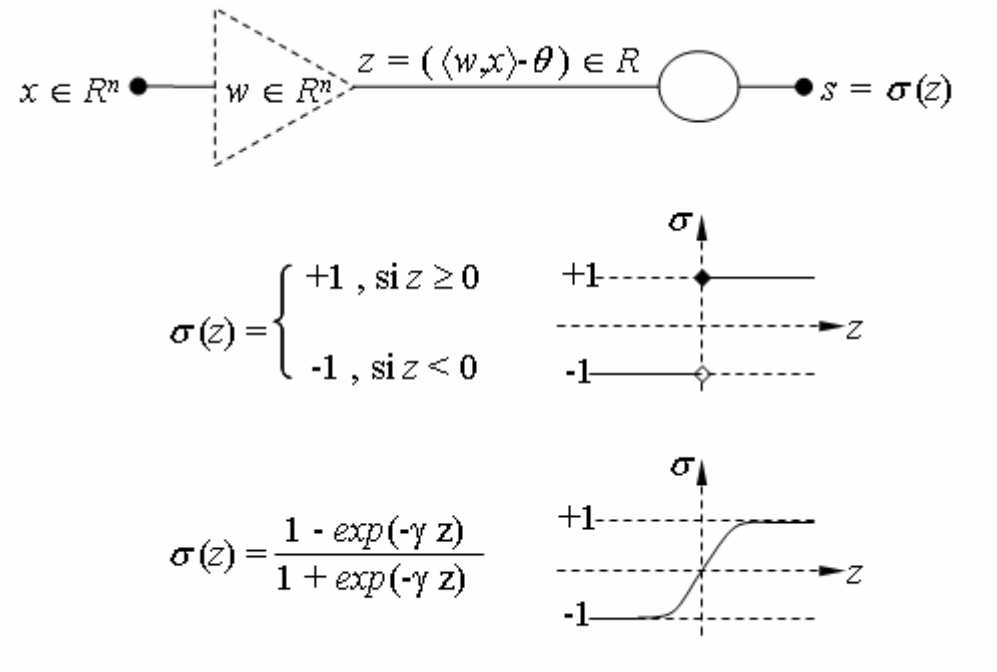

Figura 3: Perceptrón o *Lineal Threshold Unit*

LTU de clasificación, así como los dos semi-espacios generados, que idealmente contienen elementos de una sola clase. La leyenda que identifica el conjunto al cual pertenece cada punto es válida para los diagramas de todos los capítulos. Nótese que la expresión donde la suma ponderada $\langle w, x\rangle$ tiene el valor del umbral θ, corresponde al hiper-plano separador, mientras que las dos expresiones donde se cumplen desigualdades con ese umbral, corresponden a los puntos que pertenecen a los semi-espacios de cada una de las clases.

Un problema importante asociado con estos modelos, es que no se puede garantizar que el número de planos que pueden separar efectivamente un conjunto de puntos es el óptimo [Man92].

Aunque hay algoritmos de programación bi-lineal para resolver problemas de separabilidad bi-lineal [BM92c], este sigue siendo un problema NP-Completo que

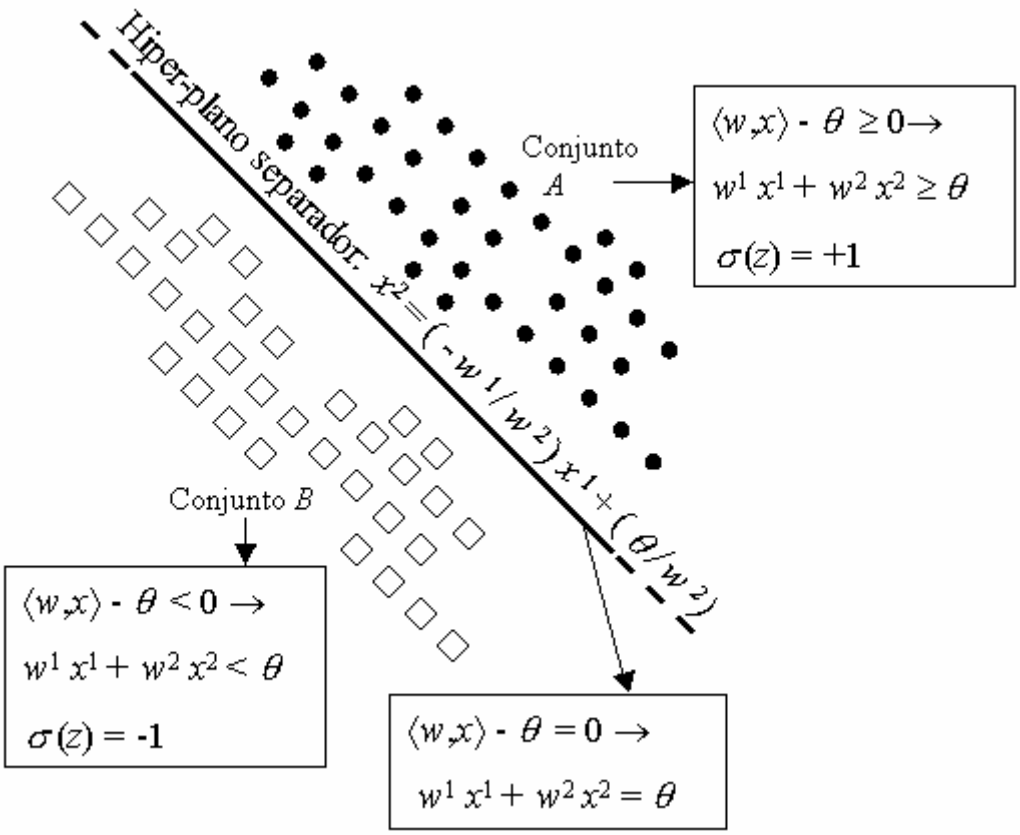

Figura 4: Hiper-plano separador generado por una LTU

sólo se ha resuelto utilizando heurísticas ¨golosas¨ (en inglés *greedy*), tales como el Método Multi-Superficies (o MSM por sus siglas en inglés), elaborado por Mangasarian [Man92], que utiliza de manera iterativa, modelos de programación lineal, para definir la superficie de separación como una función por partes. En general no se tiene certeza de cuantos planos son necesarios para separar completamente dos conjuntos de entidades.

La dificultad para encontrar el hiper-plano residía en formular un criterio que para determinar el mejor hiper-plano condujera a un modelo fácil de resolver, por ejemplo, mediante Programación Lineal. Hasta donde tenemos conocimiento, esta dificultad fue subsanada por el modelo de “Separación Robusta mediante Programación

Lineal” (originalmente bautizado *Robust Linear Programing Separation*) [BM92a], que es el utilizado por el algoritmo MSM de Mangasarian en cada iteración.

Dicho modelo obtiene un hiper-plano de separación que minimiza una media de los errores de clasificación, tomando en cuenta a cada error como el valor positivo proporcional a la distancia que separa a cada objeto mal clasificado del hiper-plano que se supone lo debería clasificar. Las distancias entre los objetos bien clasificados y el hiper-plano no son tomados en cuenta en momento alguno. Lo descrito se muestra en el Modelo 1. La representación geométrica de lo que logra el Modelo 1 se muestra en la Figura 5. Téngase en cuenta que, salvo en la sección 4, las figuras que ilustran superficies de clasificación *hipotéticas* en dos dimensiones, que sólo pretenden ilustrar geométricamente el comportamiento típico de los modelos y algoritmos a los que hacen referencia. En la sección 4, las figuras que muestran puntos y superficies en el espacio de dos dimensiones, si corresponden a los resultados numéricos *reales* de la aplicación de modelos y algoritmos.

$$
\begin{aligned}
&\min_{w,\theta,y_A,y_B}\left(\frac{\langle e, y_A\rangle}{m}+\frac{\langle e.y_B\rangle}{k}\right)\\
&sujeto\ a:\\
&\quad Bw - y_B \le e(\theta-1)\\
&\quad Aw + y_A \ge e(\theta+1)\\
&\quad y_A, y_B \ge 0
\end{aligned}
$$

Modelo 1: PL para separación robusta de Mangasarian y Bennett

En la notación escogida, m y k son el número de elementos de los conjuntos $\mathcal{A}$ y $\mathcal{B}$ respectivamente. Las matrices constantes A y B tienen m y k filas respectivamente, una fila por cada elemento a clasificar. Ambas tienen n columnas, que corresponde al número de atributos de cada elemento a clasificar, es decir, el tamaño del espacio de atributos. Los vectores variables y_{A} e y_{B} son no-negativos e indican la magnitud

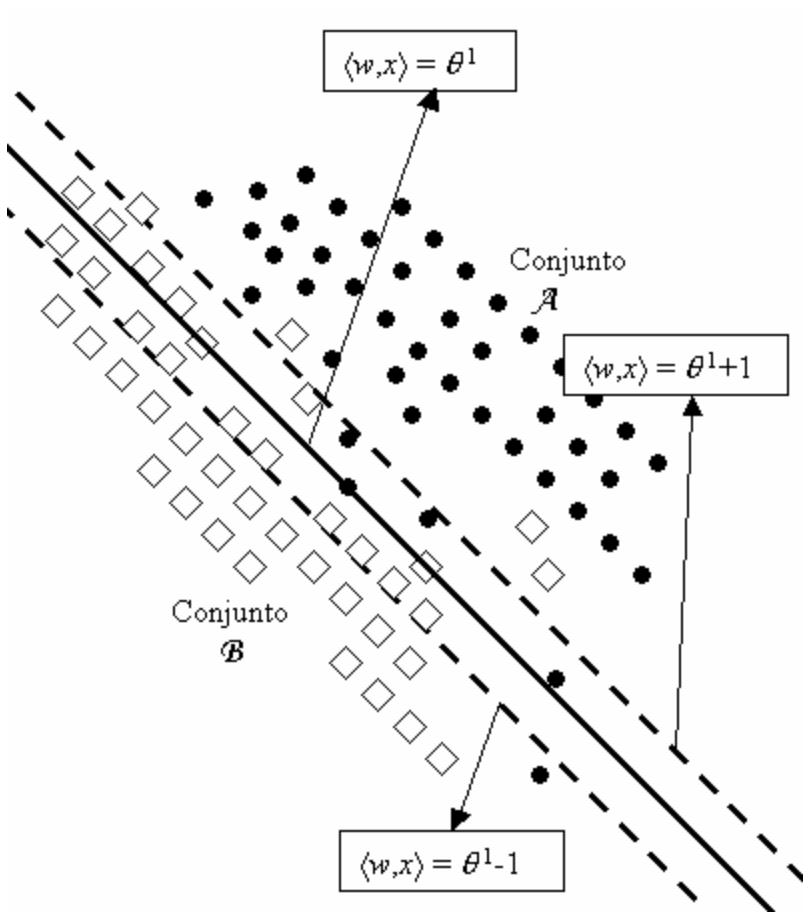

Figura 5: Separación Robusta mediante Programación Lineal

de los "errores" cometidos al clasificar cada uno de los puntos de $\mathcal{A}$ y $\mathcal{B}$ respectivamente, por esta razón sus tamaños son m y k. En el desarrollo de este trabajo se denota con e un vector de unos de tamaño apropiado. Su uso permite escribir de forma abreviada ciertas expresiones. Por ejemplo, en lugar de escribir $\sum_{i=1}^{m} y_A{}^i$, se escribe sólo $\langle e, y_A \rangle$. La variable real θ corresponde al término independiente en la ecuación del hiper-plano de separación $\langle w, x \rangle = \theta$, y que equivale al valor del umbral o frontera de decisión de la LTU que realiza la clasificación, como se indicó anteriormente. El vector variable w contiene a los "pesos" de cada una de las n dimensiones del espacio de atributos de los elementos a clasificar. También se

interpreta como el vector de los coeficientes de la ecuación del hiper-plano de separación.

Nótese en el Modelo 1, que en las primeras dos restricciones, cuando los elementos de $\mathcal{A}$ y $\mathcal{B}$ son linealmente separables, las variables y_{A} e y_{B} pueden tomar su valor mínimo de cero (0) y las restricciones son factibles. Cuando los conjuntos son linealmente inseparables, sin embargo, y_{A} e y_{B} toman un valor positivo proporcional a la distancia a la que se encuentra cada punto respecto al semi-espacio definido por el hiper-plano que corresponde a su verdadera clase.

De esta manera, el modelo trata de que en promedio haya la menor cantidad de puntos mal clasificados, y los que estén mal clasificados, estén lo más cerca posible del hiper-plano de separación.

Cuando el MSM utiliza este modelo de PL para obtener un hiper-plano, utiliza la ecuación de ese “mejor” hiper-plano, para construir otros dos hiper-planos paralelos que se trasladan hasta conseguir elementos de la misma clase hacia el mismo lado del hiper-plano. La región entre estos hiper-planos paralelos viene a ser vista como una zona “gris”, donde se encuentran elementos sin clasificar. En la Figura 6 puede observar la zona “gris” como la intersección de los semi-espacios abiertos $\langle w, x\rangle < \theta^1{}_0$ y $\langle w, x\rangle > \theta^1{}_1$.aquella que está entre los semi-espacios definidos por el hiper-plano $\langle w, x\rangle = \theta^1{}_0$, y el hiper-plano $\langle w, x\rangle = \theta^1{}_1$. Los valores de w y θ de cada hiper-plano representan a un perceptrón de la capa oculta de una red neural tipo *feed-forward* o de Re-alimentación Positiva [BM92b]. Los elementos clasificados satisfactoriamente se muestran en un tono más tenue, para resaltar los que aún quedan sin clasificar.

Una vez definidos estos hiper-planos, se toman los puntos que han quedado en la zona gris, y se les aplica nuevamente el algoritmo.

El proceso se detiene cuando se alcanza el mínimo deseado de efectividad de la clasificación empírica, o cuando se alcanza el número máximo de iteraciones. El conjunto de puntos mostrados en la Figura 6 es trabajado en 5 iteraciones hipotéticas, que generan una función clasificadora compuesta por partes lineales, que se muestra en la Figura 7 Y 8. En las Figuras 7-a, 7-b, 7-c, y 7-d, se observan las primeras 4 iteraciones.

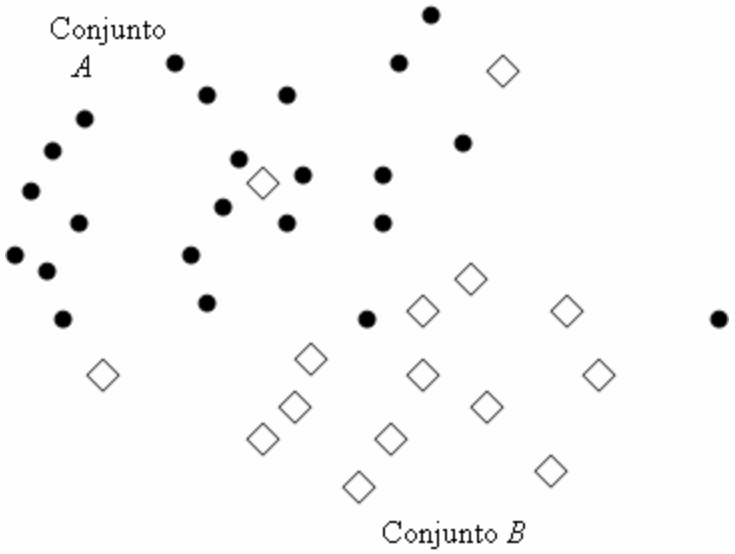

Figura 6: Ejemplo de 2 Conjuntos disjuntos de elementos en R^2

En la figura 8 se muestra la quinta iteración (y la última que se muestra para este ejemplo), que aún deja algunos puntos sin clasificar pues se trata de un caso en donde los hiper-planos generados son degenerados (ninguno de ellos clasifica punto alguno). En todas las ilustraciones las líneas en gris representan a los hiper-planos que fueron generados en iteraciones anteriores, mientras que las representadas en negro (sólidas y punteadas) representan a los hiper-planos que son “calculados” en esa iteración.

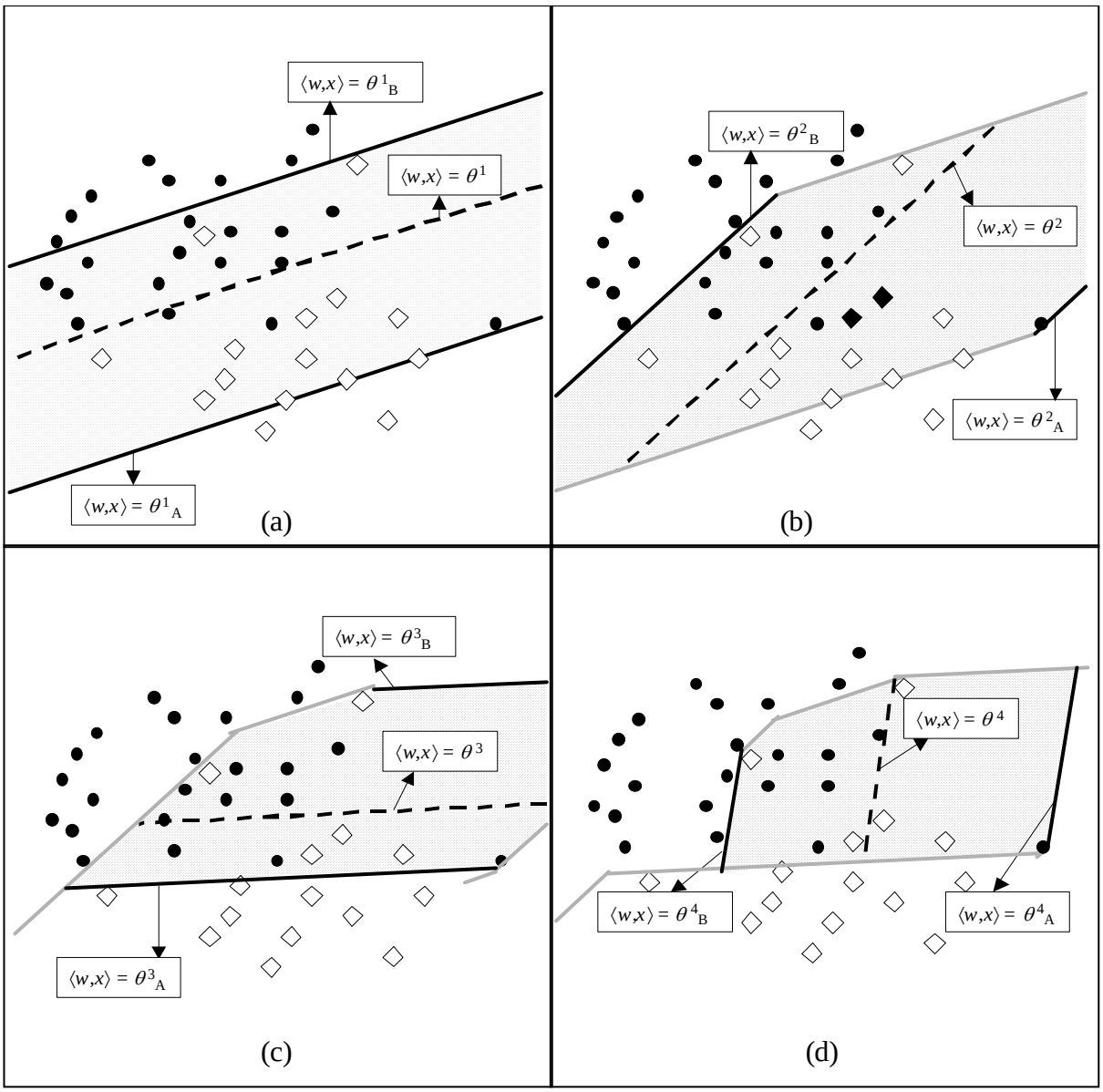

Figura 7: Iteraciones 1ra, 2da, 3ra y 4ta del MSM

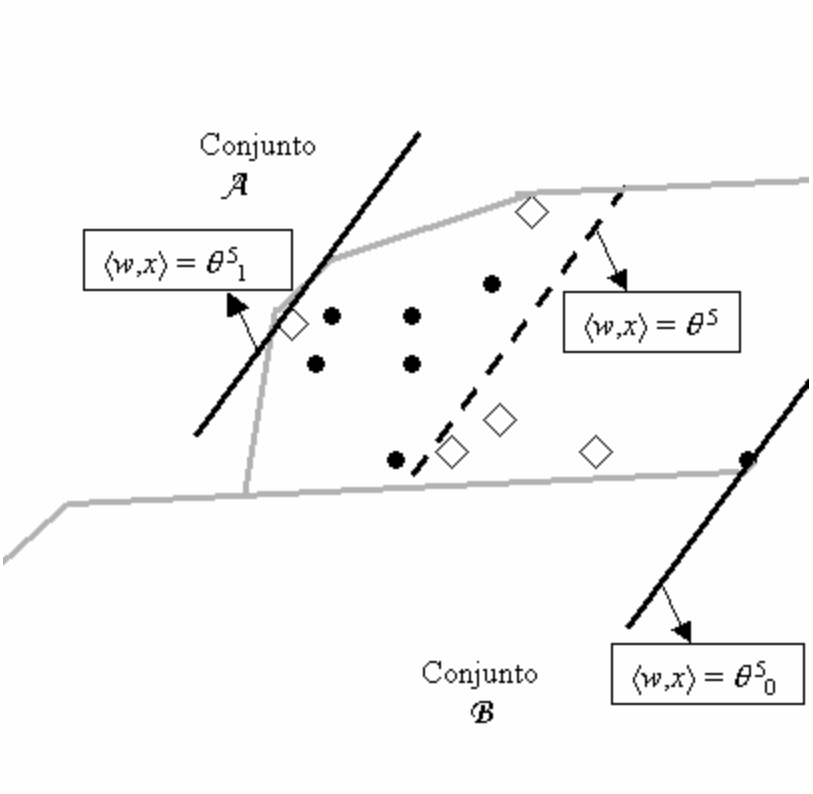

Figura 8: Iteración 5ta del MSM

1.2 Planteamiento y justificación del problema

El enfoque está basado en la optimización lineal, aplicada para obtener una sucesión de planos que se usarán para dividir el espacio de entradas a las redes neurales, en regiones poliédricas, cada una contentiva de objetos mayoritariamente pertenecientes a una categoría [Man92].

Nótese que las superficies de separación pueden ser no-lineales, mientras se mantenga linealidad en los parámetros. Esto haría que cada iteración del método fuese, esencialmente, una Máquina de Vectores de Soporte (SVM por sus siglas en inglés).

El problema que se plantea es si se puede diseñar un algoritmo que, permitiendo más flexibilidad (e inclusive más generalidad) a las ecuaciones de las superficies de separación, pueda clasificar mejor los elementos con un costo computacional comparable.

La línea de investigación que se sugiere en este trabajo, nace de una pregunta hecha durante el curso mencionado anteriormente. Ella fue: ¿es necesario que los dos hiper-planos que se generan en cada iteración, sean paralelos? En la Figura 9 se muestra el mismo caso usado como ejemplo para el MSM, pero trabajado mediante 4 iteraciones una estrategia propuesta (que no exige paralelismo a los hiper-planos de cada iteración). Nótese como mejora la tarea de clasificación, en relación con el ejemplo de 5 iteraciones del MSM, al utilizarse una menor cantidad de hiper-planos.

En la Figura 10 se observa la función de clasificación “por partes” generada por estas 4 iteraciones. Ningún punto ha quedado sin clasificar.

La respuesta a esta pregunta, para Marzo de 2004, fue “no necesariamente”. Para dar más peso a esa respuesta, o inclusive una respuesta más acertada, se comenzó la investigación que dio lugar a este trabajo.

La importancia del presente trabajo se destaca cuando se observa que hay algunos tipos de problemas para los cuales el algoritmo MSM se comporta de manera bastante ineficiente. Esto se mostrará en más detalle en el cuerpo del trabajo, en especial en el capítulo 4, en donde esto será evidenciado contundentemente por los resultados numéricos.

Se hará bastante claro que al relajar el paralelismo de los pares de hiper-planos, el problema de la clasificación binaria se resuelve de mejor manera, especialmente

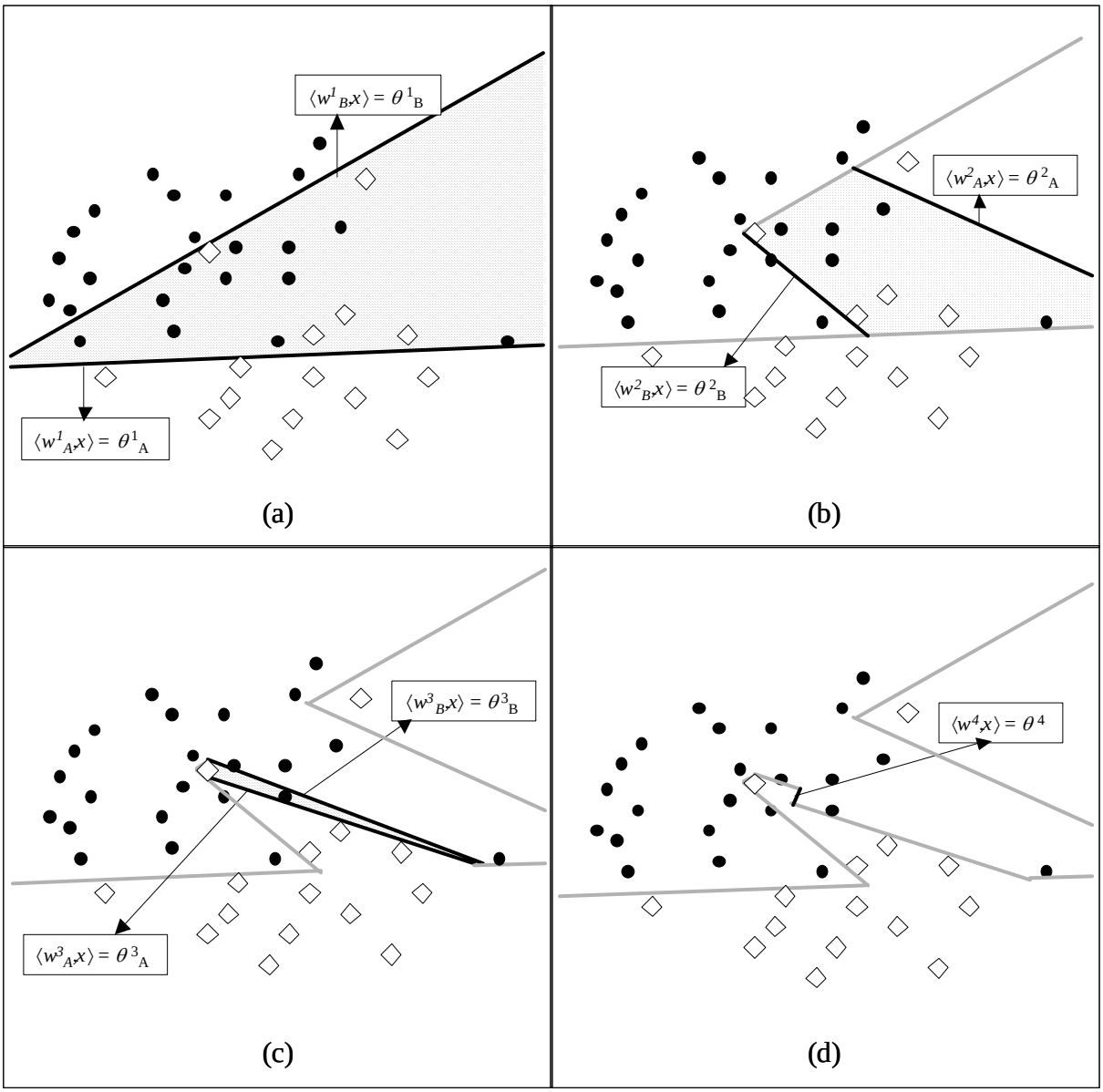

Figura 9: Clasificación Binaria con dos Hiper-planos no paralelos

cuando se desconocen las características de la estructura de la distribución de los datos. Una vez más, la navaja de Ockham parece señalar un buen camino a seguir.

Por último, cabe destacar que la investigación incita a la proposición de una familia de algoritmos que abarca al MSM como caso particular. Las pruebas numéricas que

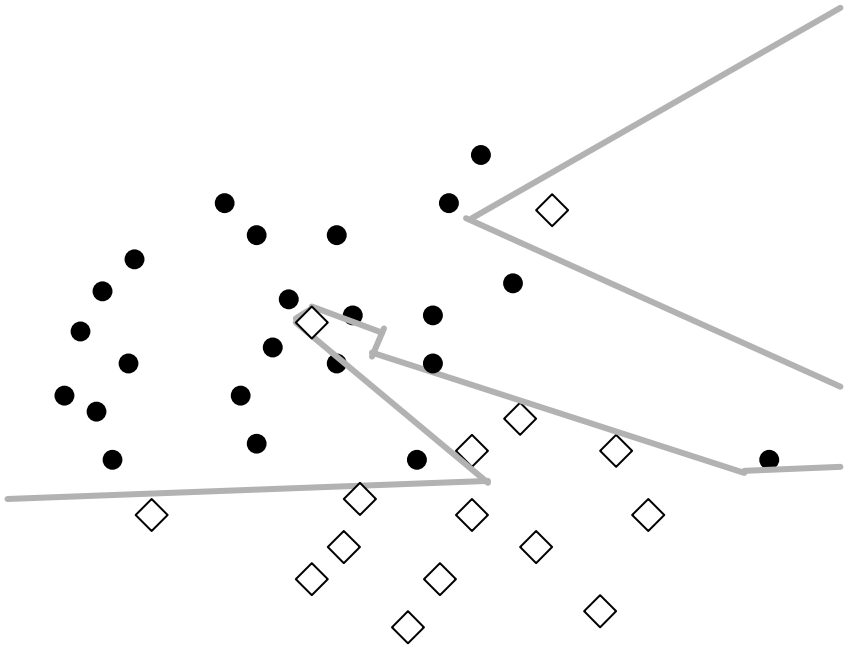

Figura 10: Superficie “por partes” generada por el algoritmo propuesto

se observarán en el capítulo 4 fueron realizadas con un algoritmo particular de esa familia, que se propone con la finalidad de garantizar un costo computacional equivalente al MSM de Mangasarian. Se observará en las experiencias computacionales que la precisión de clasificación promedio del algoritmo propuesto superó visiblemente al máximo de precisión de clasificación logrado por el MSM de Mangasarian, utilizando siempre menor cantidad de LTUs.

1.3 Sobre la organización del trabajo

En este trabajo el lector encontrará que el capítulo 2 expone lo referente al concepto de una red neural, su uso en la clasificación de entes, y nociones sobre los algoritmos clásicos para parametrizar o entrenar estas redes. En particular se

exponen las razones por las cuales se prefieren algoritmos de diseño y entrenamiento de RNAs no asociados a gradientes, como el de Bennett y Mangasarian, por sobre los más clásicos derivados del *Back-propagation* o Retro-propagación.

En el capítulo 3 se describe formalmente el problema a resolver, así como las características y bondades que son objetivo del esfuerzo de diseño que se documenta en este Trabajo de Grado. Además de la propuesta principal de un método general que funge como familia de métodos, se propone una estrategia de nomenclatura para la referencia unívoca a los algoritmos pertenecientes a esa familia. Una instancia de esa familia es generada con la finalidad de demostrar las ventajas conseguidas tan sólo considerando una relajación de las condiciones de paralelismo de los pares de hiper-planos. Se incluye una propuesta para eliminar entes clasificadores que no aporten valor al trabajo de clasificación en aras de conseguir una mejor capacidad de generalización, y una discusión sobre los criterios de optimalidad. El capítulo finaliza con una prueba de convergencia que garantiza que para la clasificación total de un conjunto de entrenamiento, se necesita siempre un número finito de iteraciones.

En el capítulo 4, se muestran los experimentos numéricos y sus resultados. Se hace énfasis en la comparación del desempeño del algoritmo propuesto en contraste con el desempeño del algoritmo de Mangasarian. Entre las consideraciones hechas en la exposición de las pruebas numéricas, está el criterio de comparación utilizado. Posteriormente se expone la interpretación de los resultados, dando pie a las conclusiones y recomendaciones. Finalmente, se encontrará un listado de la bibliografía consultada.

2. MARCO TEÓRICO

2.1 Redes Neuronales Artificiales

2.1.1 Definición

Una Red Neural Artificial, (llámese también “Red Neuronal Artificial” o “Red de Neuronas Artificiales”), puede ser descrita como una función f: $R^k \rightarrow R^l$, siendo k la dimensionalidad del vector de estímulos, y l el número de neuronas que dan la respuesta de la Red Neural Artificial. Esa función está compuesta por $n+1$ funciones más sencillas (correspondientes a cada neurona artificiales) g_i: $R^r \rightarrow S \subseteq R$, $i=0, \ldots, n$, donde r es la dimensionalidad del dominio de cada función g_i. Estas funciones de menor complejidad que se “ensamblan” para formar la Red Neural Artificial (en adelante RNA), reciben el nombre de Neuronas, y fueron concebidas inspiradas en las neuronas biológicas que conforman el tejido nervioso de los seres vivos, con el propósito de hacer mímica de los procesos neuronales que operan en ellos.

La forma en que se componen estas funciones, que pasaremos a llamar neuronas, puede ser representada por un grafo con pesos, a-cíclico y dirigido [CFP97], denotado como se muestra en la Ecuación 1:

$$\mathcal{N} = (\mathcal{V}, \mathcal{E}, w)$$

Ecuación 1: RNA denotada como grafo

donde $\mathcal{V} = \{0, 1, \ldots, n\}$ es un conjunto de $n + 1$ unidades (o neuronas), $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ es un conjunto de conexiones, y $w: \mathcal{E} \rightarrow R$ es una función que asigna un valor a cada conexión $(i,j) \in \mathcal{E}$. Para estos valores, también llamados “pesos” o “pesos sinápticos”, usualmente se utiliza una notación simplificada para $w(i,j)$, que es w_{ij}.

Cada neurona $i \in \mathcal{V}$ está asociada con su propio “campo proyectivo”

$$\mathcal{P}_i = \{ j \in \mathcal{V} : (i,j) \in \mathcal{E} \}$$

Ecuación 2: Campo proyectivo de una RNA

que representa al conjunto de neuronas alimentadas por la neurona i. También está asociada a un “campo receptivo”

$$\mathcal{R}_i = \{ j \in \mathcal{V} : (j,i) \in \mathcal{E} \}$$

Ecuación 3: Campo receptivo de una RNA

El campo receptivo $\mathcal{R}_i$ representa a las neuronas que alimentan o estimulan a la neurona i.

2.1.2 Relevancia Práctica

En la práctica, las RNAs son frecuentemente implementadas, una vez diseñadas matemáticamente, mediante hardware (equipo electrónico) cuyo comportamiento corresponde al de las funciones de transferencia con los parámetros escogidos. Al

hacerse esto, se cuenta con algunas características de cómputo que son clave para justificar la atención que se le presta a las RNA.

Entre las características de las RNAs que pretenden satisfacer necesidades no atendidas por las formas clásicas de cómputo, están [JMM96]:

- El paralelismo masivo,
- la computación y representación distribuida,
- capacidad de aprendizaje,
- habilidad para generalizar,
- adaptabilidad,
- procesamiento de información inherente al contexto,
- tolerancia a fallas, y
- bajo consumo de energía.

Estas son características que se observan recurrentemente en las estructuras neurales de los seres vivos, y aún cuando las RNAs no son, en forma alguna, el equivalente matemático de una neurona biológica, ciertamente sus características se aproximan bajo determinadas perspectivas.

Las computadoras clásicas realizan cantidades extraordinarias de operaciones secuenciales y manipulaciones simbólicas, que escapan a toda capacidad de un ser humano. Sin embargo, las RNAs, más parecidas a las neuronas biológicas que las arquitecturas de cómputo clásicas, se desempeñan de forma excelente en problemas “perceptivos” (como el reconocimiento de patrones), que resultan terriblemente complicados para los métodos clásicos de computación.

2.1.3 Usos de las RNAs

Entre las aplicaciones de las RNAs más conocidas están [JMM96]:

Clasificación de Patrones

La tarea de clasificar patrones consiste en asignar a un patrón entrante a la red (como, por ejemplo, un espectro de voz o un símbolo manuscrito) representado por un vector de características o atributos, a una o más clases pre-definidas. Existen aplicaciones de uso relativamente corriente en reconocimiento de caracteres, de voz, de ondas, de células sanguíneas, y en inspección de circuitos electrónicos impresos en tarjetas.

Agrupamiento/Categorización

En el agrupamiento, también conocido como clasificación no-supervisada de patrones, no hay datos de entrenamiento con etiquetas conocidas para las clases. Un algoritmo de agrupamiento explora la similaridad entre los patrones y coloca los patrones similares en grupos. Existen aplicaciones de uso comercial en tareas de *data mining*, compresión de información, y análisis exploratorio de datos.

Aproximación de Funciones

Supóngase un conjunto de n patrones identificados (pares entrada-salida) que han sido generados por una función desconocida (que está siendo objeto de ruido). La aproximación de funciones consiste en encontrar un estimado de esa función. Varios

problemas de modelización en la ciencia y tecnología requieren de la aproximación de funciones.

Pronósticos

De un sistema que varía su estado en el tiempo, se toman *n* muestras en diferentes momentos, generándose una secuencia, y se desea predecir, sobre la base de esa información, la evolución del estado del sistema en un momento futuro. Este tipo de usos de las RNAs es altamente pertinente y necesario para la toma de decisiones en la gestión empresarial, la ciencia y la ingeniería. La predicción de valores en la bolsa y condiciones meteorológicas son aplicaciones conocidas en este tipo de tareas.

Optimización

Una gran cantidad de problemas en matemáticas, estadística, ingeniería, ciencia, medicina y economía, se pueden entender como problemas de optimización. El objetivo de un algoritmo de optimización es encontrar una solución que satisfaga un conjunto de restricciones de tal manera que una función objetivo sea maximizada o minimizada. El Problema del Agente Viajero (TSP por sus iniciales en inglés: *Traveling Salesman Problem*), un problema NP-Completo, es un ejemplo clásico.

Control

Considérese un sistema dinámico definido por una dupla de entrada de control y la salida resultante del sistema para cada momento *t*. En modelos de RNAs adaptativas se desea generar señales de entrada de control que permitan que el comportamiento de la salida el sistema siga determinado patrón de reglas. Un ejemplo en donde este tipo de funciones es relevante, es el control del movimiento de los alerones estabilizadores de los aviones, que responden al comportamiento del sistema avión-

corrientes de aire, tratando de compensar los movimientos indeseados para mantener al avión en un nivel adecuado de estabilidad.

2.1.3 RNAs de realimentación positiva

Al tipo de RNAs, cuyos grafos carecen de ciclos, se les denomina de tipo *Feed-forward* o de Realimentación Positiva.

Cada una de las neuronas que se representan como los nodos de este grafo pesado, a-cíclico y dirigido, realiza una suma ponderada de las k entradas que provienen de su campo receptivo, que a su vez es argumento de una función que se denomina "Función de Transferencia". El dominio de esta función está acotado entre dos valores finitos y reales (salvo los casos en que la función de transferencia es lineal, que no son de interés para esta investigación). A estos valores de "salida" de las neuronas, se les suele denominar "valor de excitación" y "valor de inhibición", refiriéndose con esto a los valores "techo" y "piso" de la función de transferencia.

2.1.4 Neuronas tipo Perceptrón y Perceptrones Multi-Capa

En particular, si la función de transferencia de una neurona se toma como la función "escalón" o "umbral", o alguna función continua que emule el comportamiento de estas (como por ejemplo la tangente hiperbólica) la neurona puede ser llamada Perceptrón, o Unidad Lineal de Umbral (LTU por las siglas en inglés *Lineal Threshold Unit*) [Man92].

Cuando una RNA está compuesta únicamente por una o más neuronas tipo perceptrón, o LTUs, ciertos autores consideran correcto darle el nombre de Perceptrón *a la red resultante*. Sin embargo, para facilitar el entendimiento, se usará en este trabajo la denominación de “perceptrón” para referirse a las LTU solamente. Para hacer referencia a las composiciones de dos o más perceptrones se hablará, explícitamente, de “red de perceptrones”, “red de LTUs”, o bien “perceptrón multi-capa”.

2.1.5 El “escalón” como Función de Transferencia

Como se ha visto, las aplicaciones de las RNAs son innumerables. Sin embargo, aunque la función que genera la salida de cada neurona puede tener infinidad de formas, usualmente toma como argumento la suma ponderada que se hace a la entrada de la neurona, que conforma el “estímulo” de esa neurona.

McCulloch y Pitts [McCP43] propusieron una LTU binaria como un modelo computacional para la neurona artificial que constituye el modelo clásico de neurona en aplicaciones de clasificación binaria. El modelo de McCulloch y Pitts computa la suma ponderada de las n señales de entrada, x^j, $j=1,2,...,n$, y genera una salida unitaria positiva (+1), por ejemplo, si la suma sobrepasa un umbral θ, y una salida nula (0) en caso contrario. Estos constituyen los valores de excitación e inhibición de una LTU. Matemáticamente se escribe de la siguiente manera:

$$s = \sigma(z) = \sigma\left(\sum_{j=1}^{n} w^j x^j - \theta\right)$$

Ecuación 4: Función de transferencia de una LTU

donde σ es la función escalón centrada en el cero de la escala (0), y w^j es el "peso sináptico" asociado con la j-ésima entrada. Los pesos positivos corresponden a sinapsis excitatorias, mientras que los pesos negativos modelan a las sinapsis inhibitorias.

Los límites que acotan la función escalón pueden ser ajustados a cualquier valor mediante operaciones sencillas de escalamiento y traslación de la curva. Por esta razón, en general, los valores "piso" y "tope" son cualquier valor escogido por el modelador de forma arbitraria. En general se utilizan sigmoides para emular a la función escalón, con la forma:

$$\sigma(z) = \frac{1}{1 + \exp(-\gamma z)}$$

Ecuación 5: Sigmoide clásica como función de transferencia

o tangentes hiperbólicas (también consideradas sigmoides, recientemente) con la forma:

$$\sigma(z) = \frac{1 - \exp(-\gamma z)}{1 + \exp(-\gamma z)}$$

Ecuación 6: Tangente hiperbólica como función de transferencia

En ambos casos γ es el parámetro que define la pendiente de la función cuando el argumento de la misma es nulo.

Nótese que la diferencia entre la función de transferencia que tiene forma de sigmoide, y la que tiene forma de tangente hiperbólica, radica en que la segunda tiene un factor de escalamiento igual al doble del rango de la sigmoide, y está trasladada una unidad hacia los negativos [IG04]. En otras palabras, si la $f_1(x)$ es la función sigmoidal, y $f_2(x)$ es la función tangente hiperbólica, entonces:

$$f_2(x) = 2\,f_1(x) - 1$$

Ecuación 7: Equivalencia entre una sigmoide clásica y la tangente hiperbólica

Esta forma de modelar al perceptrón o LTU tiene la ventaja, sobre la función escalón, de ser continua y derivable en toda su extensión (aunque una pendiente γ muy elevada puede originar problemas computacionales serios, lo cual hace que el valor de la pendiente γ generalmente sea pequeño).

La utilidad que presentan las LTUs por su función de transferencia tipo escalón, es que el lugar geométrico donde ocurre la transición entre un valor y otro de la función, es un hiper-plano en el espacio de los atributos. Es decir, sea x el vector en R^n que estimula con sus n atributos a la LTU, y sean $w \in R^n$ el vector de pesos sinápticos y $\theta \in R$ el umbral de la LTU, entonces el hiper-plano $\langle x, w \rangle - \theta = 0$ separa el espacio de los atributos en dos. Uno de los semi-espacios abiertos obtenidos, $\langle x, w \rangle - \theta > 0$, corresponde a los lugares en los que la LTU respondería emitiendo una señal igual a s=+1 (o cualquiera que sea el valor “tope” de la función escalón utilizada), y el otro semi-espacio abierto, $\langle x, w \rangle - \theta < 0$, corresponde a los lugares en los que la respuesta sería s=–1 (o el valor que tenga el “piso” de la función escalón). A los puntos sobre el hiper-plano $\langle x, w \rangle - \theta = 0$ se les asigna una salida arbitraria, que posiblemente dependa de la aplicación de la red.

2.1.6 Arquitectura de los Perceptrones Multi-Capa

Los Perceptrones Multi-capa, como el nombre lo indica, poseen varias capas de neuronas. Cada una de ellas posee una o más neuronas, y se pueden clasificar en

a) capa de entrada
b) capas ocultas
c) capa de salida, tal y como se observa en la Figura 2 del capítulo 1.

Aunque la convención usual es la de llamar “capa de entrada” a unos nodos que sólo reciben las señales del exterior de la RNA y transmitirlas a las neuronas de la siguiente capa, el autor considera que dado que no existe función de transferencia en ese nivel, llamarle “capa neuronal de entrada” resulta propenso a confusiones. La capa de salida es la que da la respuesta general de la red ante cada estímulo recibido, y puede tener más de una neurona. Sin embargo, para realizar clasificación binaria es suficiente utilizar una LTU en esa capa.

Las redes de perceptrones superan muchas de las limitaciones de los perceptrones simples, aunque hubo tiempos en los que eran poco usadas porque no se disponía de un algoritmo que lograra entrenarlas de manera eficiente. Las regiones de decisión que pueden construirse con este tipo de RNAs son equivalentes a regiones poliédricas, o, lo que es lo mismo, regiones separadas por una superficie lineal por partes. A mayor cantidad de perceptrones en la red, mayor es la complejidad que puede alcanzar la superficie de decisión. Por ejemplo, un perceptrón con una capa oculta de 2 neuronas es suficiente para resolver el problema del XOR (o exclusivo), mediante dos superficies planas.

Hay autores, como R. P. Lippmann [Lip87] que han aseverado que un perceptrón de una capa oculta es incapaz de separar regiones “enredadas” o “engranadas”, es decir, no-linealmente separables. Sin embargo, en el trabajo de Mangasarian [Man92] se evidencia como un perceptrón de una capa oculta puede resolver este tipo de problemas. De hecho, un perceptrón Multi-capa de este tipo puede comportarse, si se escogen correctamente los pesos sinápticos de la red, como un árbol de decisión binario, en el que cada hoja es una función LTU. No hay par de conjuntos de datos finitos y disjuntos que no pueda ser separado por una red así, tal y como se afirma [Man92], aunque la afirmación se le atribuye Hornik et al 1989.

Esto se discutirá en más detalle cuando se exponga el trabajo de Programación Matemática en Redes Neurales realizado por Mangasarian.

Aún así, cabe resaltar que no se sugiere con esto, que se deseche, en el campo de las redes neurales para la clasificación binaria, el uso de perceptrones con más de una capa oculta. Aunque ellos constituyan separadores universales, también es cierto que un perceptrón con más capas puede realizar trabajos equivalentes con una cantidad de neuronas significativamente menor.

2.1.7 Aprendizaje de las Redes Neurales Artificiales

Para optimizar las redes de perceptrones, las funciones de transferencia se prefieren como sigmoides para utilizar métodos relacionados a gradientes, en una estrategia de “entrenamiento” que se denomina *Back-progapagation* o Retro-propagación. Resulta paradójico que de este tipo de neuronas, se desearía aproximarlas lo más posible a neuronas con la función escalón (es decir, resultaría ideal para las aplicaciones de la red, que el parámetro de pendiente fuese *muy grande*). El problema está en que mientras más se parece a la función escalón, los métodos de optimización asociados a gradiente se comportan de peor manera.

Esto puede verificarse con un ejemplo numérico pequeño. Si se toma una red con un solo perceptrón, y llamamos s^k al valor arrojado por la red (binario) para cada $x^k \in R^2$ que se le presenta como estímulo para que sea clasificado, siendo que c^k es su verdadera clasificación (expresada como binario), el problema típicamente se podría representar como:

$$\min_{w,\theta} \frac{1}{2} \sum_{k=1}^{m} \left(s^k - c^k \right)^2$$

Modelo 2: Minimización del riesgo empírico de clasificación binaria

Si la salida del perceptrón se aproxima por una sigmoide continua y derivable con un γ suficientemente grande, definida como

$$s^k = \sigma(x) = \frac{1-\exp(-\gamma.(\langle w, x\rangle - \theta))}{1+\exp(-\gamma.(\langle w, x\rangle - \theta))}$$

Ecuación 8: Salida de un perceptrón

y se utiliza un método asociado a gradientes, se obtienen resultados interesantes. Estos métodos, ya sea que en estos métodos (1) se tomen “pasos” de tamaño fijo en la dirección del gradiente después de que cada valor de los datos es analizado, o que (2) se tome un “paso” de tamaño fijo en el gradiente compuesto por los calculados para todos los valores presentes en los datos de entrenamiento, el problema dista mucho de comportarse en forma ideal.

De la primera forma, con frecuencia el método falla, y de la segunda, la cantidad de iteraciones necesarias para alcanzar la convergencia puede ser indeseablemente grande [Dix93]. Por esta razón cuando un perceptrón posee sólo una neurona o LTU, usualmente se utiliza el método de Rosenblatt, que no usa gradientes. Sin embargo, no se puede probar su convergencia si los puntos son linealmente inseparables, lo cual en general es una condición desconocida.

Sin embargo existen otras razones por las cuales métodos basados en gradientes, como la Retro-propagación, pueden resultar indeseables si se desea una alta definición en la clasificación binaria que hace cada perceptrón. En particular una buena razón es que para lograr esa alta definición, se desearía que la sigmoide tuviera un gradiente casi nulo (idealmente igual a cero) en todo el dominio de la función, salvo cuando alcanza el límite del umbral, en el cual el gradiente necesariamente es un valor finito γ. Esta propiedad, además de la no convexidad y la existencia de numerosos puntos estacionarios (ni siquiera óptimos locales), hacen

que el autor considere relevante toda línea de investigación que pretenda proponer formas de entrenar RNAs, sin el uso de gradientes.

En la actualidad, tal y como lo dice Breiman en [Bre95], hay muchas ideas contradictorias respecto a las redes neurales, como por ejemplo:

- ¿Por qué las RNAs con muchos parámetros no sobre-ajustan los datos?
- ¿Cuál es el número efectivo de parámetros?
- ¿Por qué la retro-propagación no se estanca en mínimos locales “malos” tanto como se esperaría?
- ¿En qué momento debería detenerse la retro-propagación y usar los parámetros ya obtenidos?

Luego de la realización de la presente investigación se piensa que una pregunta de igual o mayor importancia que la última mencionada arriba sería:

- ¿Por qué seguir usando retro-propagación?

En este trabajo se desea minimizar la cantidad de elementos mal clasificados por redes de perceptrones multi-capa, rigiéndose por algún criterio razonable de medición de error, sin utilizar los gradientes de las funciones de transferencia utilizadas, pues se encuentran razones para ello.

En todo caso es necesario investigar en métodos alternativos para el aprendizaje de este tipo de estructuras, para tener una “caja de herramientas” en la que se seleccionen las herramientas que se adecuen a la tarea que se enfrente. Es necesario, tal y como lo señala Wolpert en [Wol95], crear un inventario de las técnicas existentes en el aprendizaje supervisado, y por ello han de responderse preguntas tales como:

- ¿Cuáles son las fortalezas relativas de las diferentes estrategias de aprendizaje supervisado?

- ¿Cuáles son sus debilidades relativas?
- ¿En qué se diferencian sus suposiciones?
- ¿Hasta que punto se duplican unas a otras?
- ¿Qué ideas fundamentalmente distintas e intuiciones encarnan colectivamente?
- ¿Qué puede ganarse de combinar técnicas provenientes de diferentes enfoques?

Desde este punto de vista, tomando al aprendizaje supervisado como un área de la ciencia-ingeniería, queda claro que el uso ciego y sistemático de técnicas por parte de los ingenieros, sólo porque “ya funciona”, hace que esta ciencia esté encaminada sólo por las preocupaciones técnicas. Se toma un concepto de un campo (por ejemplo la retro-propagación), y observa qué problemas puede resolver. Aunque los resultados de este fenómeno son extremadamente útiles [Wol95], en cierta forma va en contra-sentido. Deberían determinarse cuáles son las preguntas o problemas de interés, y entonces proceder a darles respuesta, inventando nuevas herramientas, de ser necesario.

En este caso, se considera que aunque los métodos de gradiente pueden entrenar perceptrones multi-capa que utilicen sigmoides para emular el comportamiento de la función escalón, no significa que sean la mejor forma de hacerlo. De hecho, la suposición de que una sigmoide (como las que se necesita tener en una ARN para poder utilizar métodos de gradiente) se comporta razonablemente parecido a un escalón, no es conveniente en muchos contextos.

El enfoque debe ser, por el contrario, primero tener en cuenta cuáles son las características del problema y del objetivo que se tiene, y posteriormente buscar entre las herramientas, en base a sus potencialidades.

2.2 Perceptrones como clasificadores binarios lineales

La clasificación binaria de elementos mediante superficies lineales o multi-lineales (superficies no lineales, pero lineales por partes), es una aplicación natural para funciones como los Perceptrones y los Perceptrones Multi-capa. El problema de encontrar una o varias superficies lineales que separen dos conjuntos disjuntos no debe ser visto como un problema estrictamente asociado a las RNAs, ni a la estadística, ni a la optimización, sino que debe ser considerado un problema de "aprendizaje", en el sentido de que la(s) superficie(s) que se desean encontrar, forman parte de un espacio matemático de hipótesis, y que, en base a puntos de clasificación conocida, se desea encontrar una hipótesis que "aprenda" de esas experiencias y sea capaz de generalizar [Wol95].

2.2.1 Conjuntos Linealmente Separables e Inseparables

En la literatura existen numerosas formas de atacar el problema de encontrar un hiper-plano que separe dos conjuntos, bajo la suposición de que éste existe. Esa suposición de separabilidad, en general, puede ser arriesgada, dependiendo del contexto.

Bennett y Mangasarian han formulado un modelo de PL (que puede ser considerado un *Support Vector Machine*) que encuentra una solución a este problema, que cumple con las siguientes propiedades [BM92a]:

- Si los conjuntos son disjuntos y linealmente separables, encuentra un hiper-plano separador.
- Si los conjuntos no son linealmente separables, se obtiene un hiper-plano que minimiza alguna medida de des-clasificación de los puntos, para todos los casos posibles.

- No hay restricciones impuestas para descartar casos específicos a considerar.

Este modelo, cuya formulación es la del Modelo 1, se comporta sustancialmente mejor que otros métodos similares en problemas como el de la Base de Datos de Cáncer de Seno de Wisconsin. Sin embargo, el porcentaje de elementos mal clasificados aún está por el orden del 15-17% [BM92a].

Una pregunta relevante es ¿puede realizarse una mejora significativa al porcentaje de elementos mal clasificados, mediante la incorporación de uno o más superficies de separación?

2.2.2. Construcción de Perceptrones Binarios Multi-Capa

Una superficie compuesta por superficies lineales puede es perfectamente representable como un Perceptrón Multi-capa. Entonces habría que preguntarse cuáles son las características óptimas para un Perceptrón de este tipo, para clasificar de manera binaria dos conjuntos de datos. Nótese que un Perceptrón de una capa oculta, con una cantidad suficiente de LTUs en la capa oculta, puede separar completamente cualquier par de conjuntos disjuntos en R^n [HSW89] [MSW90].

En general, el entrenamiento de los Perceptrones Multi-capa se realiza sobre una red cuya arquitectura ha sido diseñada previamente. Este diseño abarca la determinación del tipo de neuronas y el número y disposición de las neuronas. Si bien en perceptrones multi-capa las funciones de activación son idénticas ya que cada neurona es un perceptrón, aún así hay que decidir sobre los valores “techo” y “piso”, o dicho de forma más formal, valores de “activación” e “inhibición”.

La disposición en capas de los distintos perceptrones puede venir dada por el problema, o ser elegida por el diseñador. En ambos casos se requiere conocimiento previo de la estructura y naturaleza de los datos con los que trabajará la red.

Parte de este conocimiento es sencillamente asociado al “tipo de dato” de entrada, y al “tipo de dato” que se espera como respuesta. Por ejemplo, el número de enlaces sinápticos (entradas) que tienen las neuronas de entrada, y el número de neuronas en la capa de salida, están dados por las variables que definen el problema. Sin embargo hay ocasiones en las que la dimensionalidad del problema se aumenta adrede mediante la incorporación de atributos “ficticios” que en realidad son combinaciones no-lineales de los atributos que originalmente tiene el dato que se le presenta a la red. Este tipo de decisiones amerita la realización de suposiciones acerca de la disposición “espacial” de los datos en el espacio de los atributos.

Hasta el momento no es posible determinar el número óptimo de capas y neuronas para cualquier problema [IG04]. En la práctica el ensayo y error reinan como criterio básico de diseño de las arquitecturas neuronales. Para tipos de problemas particulares existen heurísticas que permiten lograr arquitecturas que bajo cierto criterio pueden considerarse eficientes. El algoritmo que se pretende proponer en este trabajo de investigación, así como el Método Multi-Superficies de Mangasarian [Man92] constituyen heurísticas de este tipo.

Por lo general las estrategias de diseño empiezan por tener una red ya entrenada, a la cual se le realizan cambios aumentando y disminuyendo el número de neuronas ocultas y el número de capas hasta conseguir una arquitectura adecuada para el problema a resolver, en el sentido de que proporcione una solución [IG04]. Si la cantidad de neuronas en capas ocultas es óptima o no, es difícil de saber, puesto que para un mismo problema, diferentes cantidades de neuronas pueden dar resultados satisfactorios, y con frecuencia la capacidad que tiene la red para generalizar no es demasiado sensible a pequeñas variaciones en la cantidad de neuronas.

Gran parte de los trabajos de diseño automatizado de perceptrones multi-capa se basan en meta-heurísticas evolutivas tales como los Algoritmos Genéticos, que realizan búsquedas dirigidas dentro del espacio de las arquitecturas posibles, maximizando una función de aptitud que por lo general pretende aumentar junto a la capacidad de generalización que tiene la red, y/o minimizar alguna forma de error cuadrático medio [MTH89] [YL97] [GGS01].

Los algoritmos heurísticos para el diseño automatizado de este tipo de redes tienen como ventaja sobre otras formas comunes de implementación de RNAs, el hecho de que la topología no debe ser “adivinada”. Existen dos grandes clases de heurísticas para el diseño automatizado de perceptrones multi-capa, que serán descritas a continuación, de acuerdo con el enfoque de E. Amaldi y B. Guenin [AG97]:

Métodos de Reducción o Poda (Prunning)

Este tipo de heurísticas comienza con una RNA “muy grande”, es decir, con pesos y/o neuronas innecesarias. Desde este punto de vista, al tener gran complejidad, evita que, al utilizar métodos basados en el cálculo de gradientes, se tenga una alta velocidad de aprendizaje y menos chance de quedar atrapado en un mínimo local [CFP97]. Existen trabajos teóricos [Yu92] [HH91] que muestran que la cota superior de ese número “muy grande” de neuronas no es un problema complicado.

Hay métodos de Poda que reducen el exceso de pesos y nodos durante el entrenamiento [Hint89], y métodos en los que el entrenamiento y la poda son realizados en fases distintas [SD88] (lo cual es mejor en términos de generalización [BFOS84]). Otros reducen iterativamente las neuronas escondidas y ajustan los parámetros de las neuronas restantes, vigilando que se cumple con el comportamiento entrada-salida deseado [CFP97].

Métodos Constructivos

Son utilizados para hallar los valores de los parámetros de la red, y variar su arquitectura o topología. Por esa razón no solo son considerados métodos de entrenamiento o diseño, sino de ambos. Por lo general comienzan con una red que no es suficientemente compleja para llevar a cabo la tarea, y los perceptrones se agregan en alguna forma, de manera estrictamente incremental, hasta que la red cumple con lo deseado.

Aunque la capacidad de generalización de la red no es altamente sensible a variaciones pequeñas en el tamaño de una red, si las neuronas que se agregan o eliminan están en una capa oculta, el tamaño de una red siempre tiene efecto en la capacidad de generalización. Siguiendo el principio de la Navaja de Ockham, entre todas las redes capaces de realizar una tarea, las más sencillas (con el menor número de grados de libertad) son las que tienen una mayor capacidad de generalización. E. Amaldi y B. Guenin [AG97] señalan los estudios de Baum y Haussler [BH89], así como los de Cohn y Tesauro [CT92], que hasta el momento validan este principio.

Diseñar redes lo más compactas posibles es una tarea tan o más difícil que la de encontrar un conjunto de valores para los pesos y umbrales de la arquitectura de una sola de esas redes, lo cual ya es un problema NP-completo [BR89] [Judd90].

Sin embargo, dada la baja sensibilidad de la capacidad de generalización de una red, ante las variaciones pequeñas del número de neuronas, el interés no es obtener un óptimo de la cantidad de neuronas, sino acercarse razonablemente a ese número.

La ventaja de utilizar heurísticas constructivas para diseñar RNAs está en que van de lo más sencillo a lo más complejo, por lo que tienen menores requerimientos computacionales que las heurísticas que empiezan con modelos más complejos de lo necesario, y se dedican a reducirlos. Esto hace a estas estrategias atractivas desde el punto de vista de *hardware*, aunque trae como consecuencia, que, por lo general, las neuronas no se entrenan de forma paralela, ya que se entrenan a medida que se insertan a la red, de forma incremental.

Existen muchos problemas prácticos que han sido resueltos siguiendo este tipo de heurísticas de manera exitosa. Véanse por ejemplo [BGP91], [CP95], [Man92], [MG93] [FL90], [Gall86], [M y [SN90]. El interés en este tipo de trabajos está siendo estimulado por el aumento en la necesidad de trabajar con grandes cantidades de información. Recuérdese que mientras se cuente con más información para “construir” una función que resuelva nuestro problema, el riesgo empírico será más próximo al riesgo real de clasificación.

Finalmente, acerca de las heurísticas constructivas para diseño de RNAs, se debe decir que en su gran mayoría están basadas en estrategias “golosas” (del término *greedy* en inglés), ya que buscan, en cada iteración, resolver el problema de la mejor manera. Aún así, existen algunos métodos no-golosos, que aunque resuelven el problema, resultan ser algoritmos de alta complejidad computacional. Como ejemplo pueden tomarse los trabajos realizados en [BBlue96], [BBred94].

Entre las estrategias golosas, hay las que generan una red de una capa oculta, en donde se adicionan en la capa oculta en cada iteración, cierta cantidad de neuronas. Otras estrategias son capaces de generar capas nuevas en la red.

2.2.3 Método Multi-Superficies de Mangasarian

El Método Multi-Superficies de Mangasarian, denotado por su autor en [Man92] como MSM (por sus siglas en inglés *Multi-Surface Method*) es un método constructivo goloso que resuelve en cada iteración un problema de programación lineal robusto para separación de dos conjuntos mediante un hiper-plano. En particular, el modelo utilizado es el propuesto por Mangasarian y Bennett en [BM92a]. Luego de resuelto el modelo PL, el algoritmo genera dos planos paralelos, entre los cuales quedan los elementos que no se pueden clasificar, y que pasan a la siguiente iteración en la que (de ser necesario) se agrega un nuevo par de planos paralelos.

Se mostrará un ejemplo *hipotético* del MSM. Supóngase que se tienen datos pertenecientes a dos conjuntos disjuntos $\mathcal{A}$ y $\mathcal{B}$ dispuestos en R^2 tal y como se muestra en la Figura 11, y se aplica sobre ellos el algoritmo MSM.

La función lineal por partes que separa ambos conjuntos fue formada en tres (3) iteraciones. Primero, mediante un PL para separación lineal robusta [BM92a], se obtiene un hiper-plano $\langle w^1, x \rangle = \theta^1$ que separa “de la mejor manera” ambos conjuntos. A partir de ese hiper-plano se construyen:

a) El hiper-plano $\langle w^1, x \rangle = \theta_A{}^1$, desplazándolo hacia el conjunto $\mathcal{B}$ de tal forma que hacia ese semi-espacio clasifica elementos del conjunto $\mathcal{B}$ cometiendo pocos

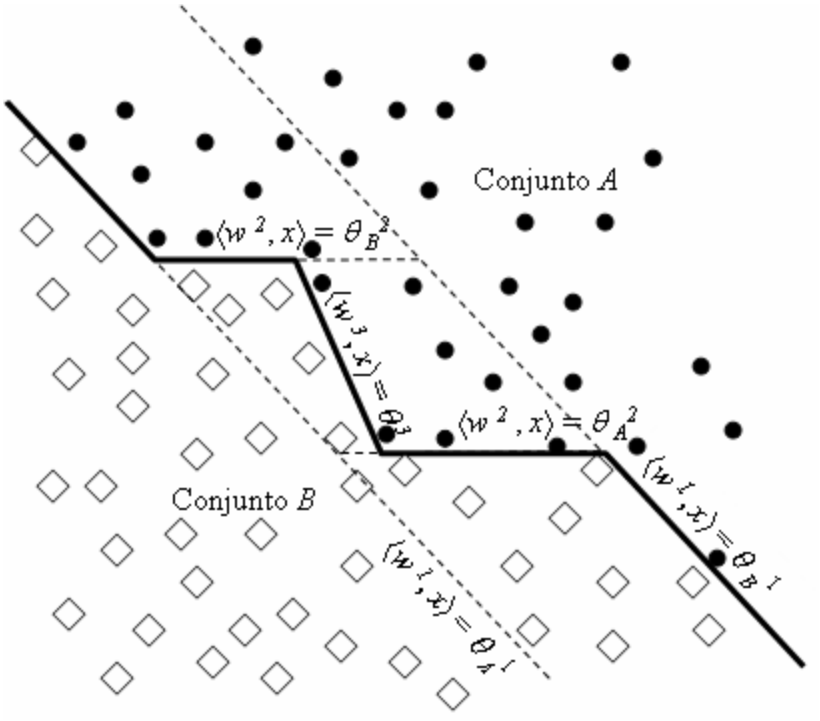

Figura 11: Tres iteraciones del MSM

errores de clasificación (o ninguno). Esto se consigue desplazando el hiper-plano cambiando el valor de θ hasta que pasa el último elemento de $\mathcal{A}$ que se encuentra hacia ese semi-espacio.

b) El hiper-plano $\langle w^1, x\rangle = \theta_B^{\ 1}$, desplazándolo hacia el conjunto $\mathcal{A}$ de tal forma que contiene de ese lado del semi-espacio, mayormente elementos de $\mathcal{A}$. Esto se logra cambiando el término independiente de la ecuación del hiper-plano, de manera análoga al hiper-plano anterior, pero eliminando los errores de clasificación del otro conjunto.

El resultado es que $\theta_A^{\ 1} < \theta_B^{\ 1}$ y el semi-espacio $\{x \mid \langle x, w^1\rangle > \theta_B^{\ 1}\}$ contiene sólo puntos de $\mathcal{A}$, mientras que el semi-espacio $\{x \mid \langle x, w^1\rangle < \theta_A^{\ 1}\}$ contiene sólo puntos de $\mathcal{B}$. Los puntos separados en estos semi-espacios son descartados y el proceso se repite en

la segunda iteración para los puntos que se encuentran entre los planos $\langle w^1, x\rangle = \theta_A^{\ 1}$ y $\langle w^1, x\rangle = \theta_B^{\ 1}$, obteniendo de esa manera los planos paralelos $\langle w^2, x\rangle = \theta_A^{\ 2}$ y $\langle w^2, x\rangle = \theta_A^{\ 2}$. Estos últimos también separan con poco error (o sin errores) puntos de $\mathcal{A}$ y de $\mathcal{B}$ respectivamente, que se descartarse nuevamente, dejando los puntos que se encuentran entre los dos últimos hiper-planos para la tercera iteración. Al ejecutarse ésta, se consigue que el plano $\langle w^3, x\rangle = \theta^3$ baste para separar los puntos restantes.

La función lineal por partes que separa los datos del ejemplo de la Figura 11, se describe como un árbol de decisión en la Figura 12 [Man92].

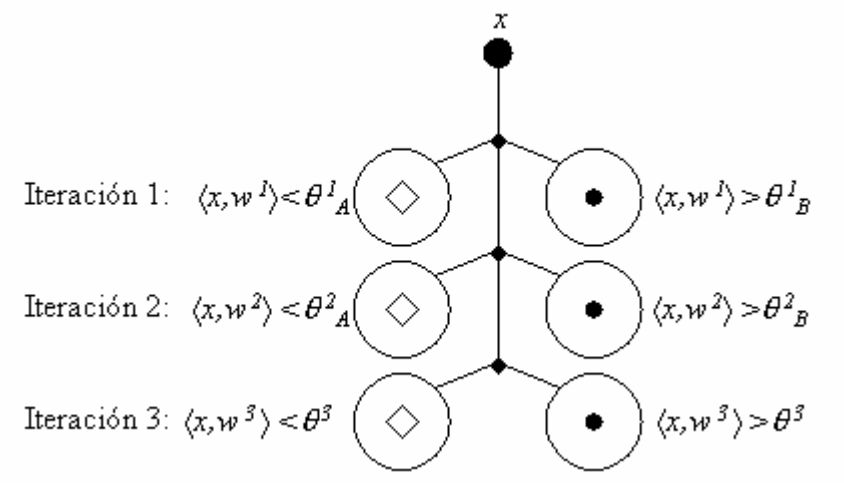

Figura 12: Árbol de Binario correspondiente al método MSM

En la Figura 13 se muestra, tal y como lo presenta Mangasarian en [Man92], una representación de este árbol de decisión como una RNA de una capa oculta, cuyas LTUs cumplan con que la convención

$$s_i(x_i) = \sigma(\langle w, x_i\rangle - \theta) = \begin{cases} 1 \; si \; \langle w, x_i\rangle \geq \theta \\ 0 \; si \; \langle w, x_i\rangle < \theta \end{cases}$$

Ecuación 9: Convención de la respuesta de las LTUs del MSM

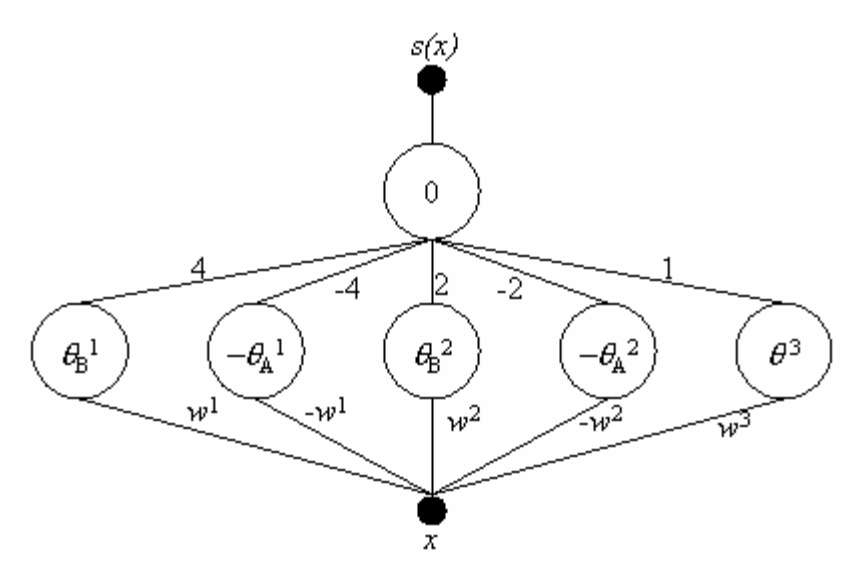

Figura 13: RNA generada por el MSM

Nótese que esta convención es distinta a la que se usará en la propuesta de esta investigación, en la que el valor de inhibición de las neuronas es s=–1.

El primer par de LTU de la capa escondida, con sus respectivos arcos de entrada, representan a los planos $\langle w^1, x\rangle = \theta_A{}^1$ y $\langle w^1, x\rangle = \theta_B{}^1$, Debido a que $\theta_A{}^1 < \theta_B{}^1$, sólo una de las neuronas puede ser activada por un x específico. Cuando la LTU1 dispara, la LTU2 no lo hace, y como 4>2+1, se puede verificar que $x \in \mathcal{A}$ sin importar lo que las otras LTUs hagan. Similarmente cuando LTU2 dispara, sin lugar a dudas $x \in \mathcal{B}$. Cuando ninguna de estas dos neuronas dispara, la decisión es relegada a las LTU3 y LTU4, que determinarán de manera análoga si el individuo es de $\mathcal{A}$ o de $\mathcal{B}$. Si ninguna de estas dispara, entonces la decisión final es dejada en manos de la LTU5. Nótese que es posible que $\theta_A{}^1 = \theta_B{}^1$, en cuyo caso las primeras dos neuronas pueden disparar al unísono y por ende cancelarse mutuamente, en cuyo caso la decisión pasa a las siguientes dos neuronas. Como puede verse, lo mismo ocurre con LTU3 y LTU4.

Como puede verse, este método constructivo de diseñar la arquitectura de un perceptrón multi-capa presenta las ventajas asociadas a este tipo de métodos de entrenamiento reconocidas hasta el momento, tales como [RWW94]:

- Se traducen en la confección de redes más pequeñas.
- Aumento en la capacidad de generalización, debido a que resulta menos sensible al ruido.
- A partir de los pesos y umbrales es fácil deducir el comportamiento de la red.
- Se pueden esbozar analogías con técnicas de clasificación de patrones tales como (1) árboles de decisión y (2) sistemas expertos.

Más aún, el método construye un perceptrón de una sola capa oculta, en donde los pesos y el umbral de la última capa no son incógnitas. También se puede apreciar que, con la posible excepción de la(s) neurona(s) generada(s) en la última iteración del proceso, cada par de neuronas poseen el mismo peso (en valor absoluto, pero el signo tampoco es una incógnita. En otras palabras, las neuronas están agrupadas en pares que comparten los pesos y umbrales, reduciendo así la complejidad de la red, respecto a una de arquitectura equivalente que fuese entrenada a la manera tradicional (usando los gradientes). Esto, tal y como lo apuntan Hinton y Nowlan en [HN95], hace que haya menos información en los parámetros de la red, lo cual resulta en una mejora en la capacidad de generalización. De hecho, se sabe [HN95], que el tomar grupos de neuronas forzadas a compartir un mismo peso es una técnica que en muchos casos resulta efectiva para lograr una buena generalización.

En el Algoritmo 1 se especifica formalmente el algoritmo MSM, omitiendo procedimientos de anti-degeneración, que se usan cuando ocurre que algún par de hiper-planos no logra discriminar dato alguno, tal que $\mathcal{A}^{i+1} = \mathcal{A}^{i}$, y $\mathcal{B}^{i+1} = \mathcal{B}^{i}$, para algún i de la notación del algoritmo. Téngase en cuenta que $\mathcal{A}$ y $\mathcal{B}$ son conjuntos de puntos disjuntos en R^n representados por las matrices A y B, de dimensiones mxn y kxn respectivamente. El modelo de PL de separación lineal robusta de dos conjuntos

Procedimiento: Método Multi-Superficies

Comienzo

Paso 1: Inicialización

a) Inicialización del nivel del árbol (i:=0);
b) Inicialización del conjunto $\mathcal{A}$ de ese nivel ($\mathcal{A}^0$:=$\mathcal{A}$);
c) Inicialización del conjunto $\mathcal{B}$ de ese nivel ($\mathcal{B}^0$:=$\mathcal{B}$);

Paso 2: Procedimiento a ejecutar en el nivel i del árbol

Si ($\mathcal{A}^i \approx \emptyset$) **ó** ($\mathcal{B}^i \approx \emptyset$) **Entonces**

Parar

En caso Contrario

a) Hallar los parámetros de las superficies de separación resolviendo $LP(A^i, B^i)$.
$(w^i, \theta^i, y^i_A, y^i_B) \in arg\ min\ (LP(A^i, B^i))$
b) Definir $\theta^i_A := \min_j A^i_j w^i$, $\theta^i_B := \max_j B^i_j w^i$
c) Definir los descendientes de $\mathcal{A}^i$ y $\mathcal{B}^i$:
A^{i+1}:={ $A^i_m \mid A^i_m w^i \leq \theta^i_B$}, B^{i+1}:={ $B^i_m \mid B^i_m w^i \geq \theta^i_A$}

Fin de Condicional

Paso 3: Actualizaciones para próxima iteración/nivel del árbol

a) Actualización del nivel del árbol (i:=i+1);
b) **Ir a** Paso 2

Fin

Algoritmo 1: MSM de Mangasarian

propuesto en [BM92a] se denota como $LP(A,B)$ y su conjunto solución se denota como $arg\ min\ LP(A,B)$.

Nótese que los hiper-planos $\langle w^i, x \rangle = \theta_A{}^i$ y $\langle w^i, x \rangle = \theta_B{}^i$, son definidos de tal manera que siempre el hiper-plano $\langle w^i, x \rangle = \theta_A{}^i$ pasa exactamente por uno de los individuos del conjunto $\mathcal{B}$, y el plano $\langle w^i, x \rangle = \theta_B{}^i$, de igual forma, contiene al menos a uno de los individuos del conjunto $\mathcal{A}$. Es bien conocido en el área de SVM (Support Vector Machines) que para optimizar la capacidad de generalización de una superficie

clasificadora de dos conjuntos separables, debe maximizarse la distancia que hay entre los puntos de ambos conjuntos a clasificar, y la superficie clasificadora.

Para comprender este concepto, puede imaginarse dos conjuntos de puntos disjuntos separados por un margen de cierto espesor. La forma más intuitiva de separar ambos conjuntos, es pasar un hiper-plano que pase por esa zona o margen de separación, equidistante a ambos conjuntos. Colocar el hiper-plano demasiado cerca de uno de los conjuntos haría más probable que si viene un nuevo dato de ese conjunto, ligeramente más cercano al conjunto opuesto, sea mal clasificado.

Teniendo esto en cuenta, resulta claro que el hecho de que en el MSM haya individuos pertenecientes a cualquiera de los conjuntos que se están clasificando, que estén sobre la superficie, es una condición altamente indeseable en términos de su efecto sobre la capacidad de generalización. Esta constituye, a juicio del autor, una evidente oportunidad de mejora al algoritmo MSM, que en principio puede tener una solución bastante sencilla. Basta con incorporar en el algoritmo un procedimiento que desplace la superficie hasta la distancia media entre el punto perteneciente al conjunto que se desea separar, que queda sobre la superficie, y el punto que, estando en el otro semi-espacio, esté más cerca de la superficie generada. Sin embargo, las propuestas realizadas en la presente investigación van mucho más allá de esta mejora puntual al algoritmo citado. La naturaleza de las propuestas hace innecesario el realizar operaciones como la utilizada en el MSM para conseguir θ^{i}_{A} y θ^{i}_{B}. La intención del comentario es proporcionar, dentro de este marco teórico, un conocimiento crítico básico sobre algunas de las propiedades del método MSM.

Finalmente, para culminar con la descripción del MSM, en el Algoritmo 2 mostrado a continuación, se expone la estrategia propuesta por Mangasarian cuando se cumple que $\mathcal{A}^{i+1} = \mathcal{A}^{i}$, y $\mathcal{B}^{i+1} = \mathcal{B}^{i}$ para algún i:

Procedimiento para Degeneraciones para el Algoritmo MSM

Para insertarse antes del paso 3 del algoritmo MSM (Algoritmo 1)

Comienzo

Paso 3.1: Conseguir planos no degenerados

Si ($\mathcal{A}^{i+1} = \mathcal{A}^{i}$) **y** ($\mathcal{B}^{i+1} = \mathcal{B}^{i}$) **entonces**

Encontrar un nuevo w^i, θ^i_A, θ^i_B satisfaciendo:

a) $A^i w^i \geq e\theta^i_A$,

b) $B^i w^i \leq e\theta^i_B$,

c) $\theta^i_A < \theta^i_B$, **y**

d) $A^{i+1} \neq A^i$, **o** $B^{i+1} \neq B^i$

Fin de Condicional

Paso 2: continuar con el algoritmo MSM

Fin

Algoritmo 2: Procedimiento de Anti-degeneración del MSM

Se puede ver que la búsqueda de un nuevo w^i, θ^i_A, θ^i_B puede realizarse de varias maneras. La más simple, propuesta en [Man92], es construir un plano $\langle w^i, x \rangle = \theta_B^1$ que separe uno o más puntos de $\mathcal{A}^i$ y ningún punto de $\mathcal{B}^i$, o construir un plano $\langle w^1, x \rangle = \theta_A^1$ que separe uno o más puntos de $\mathcal{B}^i$, pero ninguno de $\mathcal{A}^i$.

3. MÉTODOS Y PROCEDIMIENTOS UTILIZADOS

3.1 Planteamiento del Problema

Dado el método multi-superficies propuesto por Mangasarian en [Man92], se cuestiona la justificación para exigir paralelismo a los hiper-planos generados en cada iteración. El principal argumento para esta característica de las superficies generadas es que las mismas son construidas a partir del “mejor” de los hiper-planos que pueda separar los puntos por clasificar.

Sin embargo, a juicio del autor, ese argumento no resulta enteramente satisfactorio ya que el hiper-plano a partir del cual se construyen los dos hiper-planos de cada iteración, es conseguido mediante un modelo de Programación Lineal, propuesto en [BM92a] que fue mostrado en la introducción con el nombre de Modelo 1:

$$\min_{w,\theta,y_A,y_B} \left\{ \frac{\langle e, y_A \rangle}{m} + \frac{\langle e, y_B \rangle}{k} \right\}$$
$$Sujeto\ a:$$
$$Aw + y_A \geq e(\theta + 1)$$
$$Bw - y_B \leq e(\theta - 1)$$
$$y_A, y_B \geq 0$$

Modelo 1: PL para separación robusta de Mangasarian y Bennett

Como se puede observar en la formulación del modelo lineal los errores minimizados son los correspondientes a elementos de ambos conjuntos, con *un solo hiper-plano*.

Sin embargo, lo que se desea de los dos hiper-planos generados en cada iteración del método multi-superficies, realmente, es que cada uno de los *dos hiper-planos* defina un semi-espacio que contenga *sólo elementos de un conjunto*. Adicionalmente no hay ningún interés particular en que los hiper-planos sean paralelos.

Por estas razones se considera necesario formular un método alternativo que considere lo que verdaderamente se quiere de los hiper-planos. En particular, la formulación debería explícitamente conseguir dos hiper-planos, que cometan, cada uno de ellos, errores de un solo tipo. Ese error, que corresponde a cada uno de los conjuntos, debe ser minimizado por el programa matemático.

En el presente proyecto se propone una estrategia general para resolver el problema de encontrar la multi-superficie que separe mejor dos conjuntos de puntos disjuntos. Al llevar esa estrategia a la práctica implementando un método comparable con el Multi-Superficies de Mangasarian, se definió una formulación que resulta semejante a la resolución del problema de PL robusto [BM92a] que se resuelve en cada iteración del MSM [Man92], cuando se le añade un grupo de restricciones (incluyendo el procedimiento para construir el par de hiper-planos de cada iteración). Dichas restricciones *son aquellas necesarias para exigir paralelismo* a los dos hiper-planos generados. Esto será discutido más adelante en el capítulo, cuando se presente formalmente el ejemplo de aplicación de la estrategia propuesta en la sección 3.4.

Si se consideran dos problemas de optimización idénticos salvo por el hecho de que uno de ellos presenta una o varias restricciones adicionales, aquel que tiene más restricciones obtendrá un valor óptimo *a lo sumo* igual al valor óptimo que se consigue con el menos restringido (típicamente el valor obtenido es peor). Por ende resulta claro que para lograr mejores resultados es importante eliminar la mayor cantidad de restricciones innecesarias. Por ello, si no hay algún interés pragmático en que los hiper-planos sean paralelos a pares, se debe procurar que la formulación

evite mantener esta restricción, que forma parte del método multi-superficies de Mangasarian.

Hay una consideración adicional que hacer, y es que los hiper-planos construidos a partir del “mejor” separador lineal de ambos conjuntos para los puntos a clasificar, tienen una “distancia” nula respecto por lo menos uno de los puntos clasificados por cada hiper-plano generado. Esto no es una condición deseable cuando se busca generar una superficie de clasificación que cumpla con criterios de generalidad. Esto se puede verificar en el apartado del capítulo anterior dedicado a la exposición del método de multi-superficies de Mangasarian.

3.2 Planteamiento de la Propuesta del Método General

En principio, se propone un método cuya estrategia pretende responder, de la manera más flexible y genérica posible, a las inquietudes expuestas en la sección anterior y que de hecho, contiene como caso particular al algoritmo multi-superficies de Mangasarian [Man92]. Debe notarse que, al igual que éste, el método propuesto involucra el equivalente a la creación de un árbol de decisión binaria. La similitud que tiene el algoritmo general propuesto con el algoritmo de Mangasarian resulta evidente, salvo en el hecho de que se generalizan las características del par de superficies generadas. La estrategia de este método, a la que se bautiza “Optimización Generativa de Multi-Superficies” (por razones que serán explicadas en la sección 3.3) está resumida en los siguientes pasos:

Procedimiento: Optimización Generativa de Multi-Superficies

Comienzo

Paso 1: Inicialización

a) Inicialización del nivel del árbol (i:=0);

b) Inicialización del conjunto $\mathcal{A}$ de ese nivel ($\mathcal{A}^0$:=$\mathcal{A}$);

c) Inicialización del conjunto $\mathcal{B}$ de ese nivel ($\mathcal{B}^0$:=$\mathcal{B}$);

Paso 2: Procedimiento a ejecutar en el nivel i del árbol

Si ($\mathcal{A}^i \approx \emptyset$) **ó** ($\mathcal{B}^i \approx \emptyset$) **ó** (alguna otra condición opcional que proporcione indicios numéricos de que se está perdiendo generalidad de clasificación) **Entonces**

Parar

En caso contrario

a) Hallar los parámetros de las superficies de clasificación $\varphi_A(x)=\theta_A$ y $\varphi_B(x)=\theta_B$ (tales que clasifiquen correctamente la totalidad de los elementos pertenecientes a $\mathcal{A}$ y $\mathcal{B}$ respectivamente, minimizando la cantidad de elementos de $\mathcal{B}$ mal clasificados por la superficie $\varphi_A(x)=\theta_A$, y minimizando la cantidad de elementos de $\mathcal{A}$ mal clasificados por la superficie $\varphi_B(x)=\theta_B$)

b) Descartar los elementos de $\mathcal{B}$ que hayan sido bien clasificados por $\varphi_A(x)=\theta_A$, y los elementos de $\mathcal{A}$ que hayan sido bien clasificados por $\varphi_B(x)=\theta_B$.

c) Obtener A^{i+1} y B^{i+1} (los descendientes de A^i y B^i), actualizando las matrices eliminando las filas correspondientes a los elementos descartados en cada conjunto durante el paso 2.a.

Fin de condicional

Paso 3: Actualizaciones para próxima iteración/nivel del árbol

a) Actualización del nivel del árbol (i:=i+1);

b) **Ir a** Paso 2

Fin

Algoritmo 3: Método General Propuesto

Se puede ver que el algoritmo multi-superficies de Mangasarian [Man92] es un caso particular de este método, en el cual las superficies $\varphi_A(x)=\theta_A$ y $\varphi_B(x)=\theta_B$ son los hiper-planos generados a partir del “mejor” hiper-plano obtenido mediante el método robusto de Mangasarian y Bennett [BM92a] y se generan simultáneamente.

En la Figura 14 se puede ver un conjunto de puntos sobre los cuales se aplicará el método, utilizando superficies curvas genéricas, para evidenciar el hecho de que resulta innecesaria la exigencia del paralelismo de las superficies a utilizar.

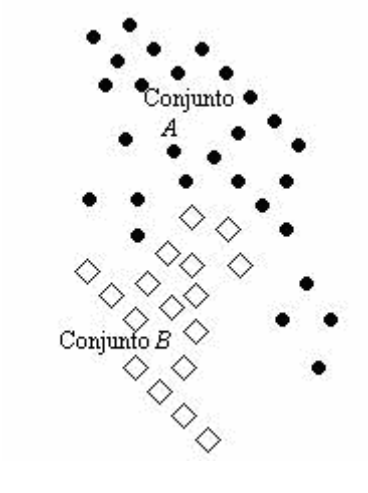

Figura 14: Ejemplo de dos conjuntos disjuntos

En la Figura 15 se observa como una superficie $\varphi_A=\theta_A$ que cumple con las condiciones expresadas en el método, al minimizar la cantidad de puntos de $\mathcal{B}$ mal clasificados y contener por completo a los puntos de $\mathcal{A}$, permite clasificar un conjunto constituido solamente por puntos de $\mathcal{B}$. Esta superficie, que en principio lo que hace es contener por completo al conjunto de puntos de $\mathcal{A}$ que hay en ese nivel del árbol, realmente sirve para conseguir puntos de $\mathcal{B}$ que pueden descartarse (por estar bien clasificados) para la próxima iteración.

Análogamente, en la Figura 16 se observa la superficie que contendrá a los puntos de $\mathcal{B}$ presentes para esa iteración, por completo. También allí se observa como esa superficie permite aislar un conjunto de puntos que son del conjunto $\mathcal{A}$ únicamente.

Las Figuras 17 y 18 ilustran el resultado del método en esta iteración de ejemplo. Nótese que el método, como estrategia general de generación de superficies, no exige condiciones especiales a las superficies. De hecho, los parámetros de ambas superficies pueden hallarse de manera independiente. Entre otras cosas podría obtenerse cada superficie mediante procesamiento paralelo, si a cada una de las computadoras se le proporciona el conjunto de puntos a resolver en cada iteración. Sin embargo, en el presente trabajo se realizará un planteamiento más simple de este algoritmo, comparable con el MSM de Mangasarian [Man92]. Con esta finalidad se exige linealidad en los parámetros a las superficies. Esto permite el uso de la Programación Lineal o Cuadrática, con extrema facilidad.

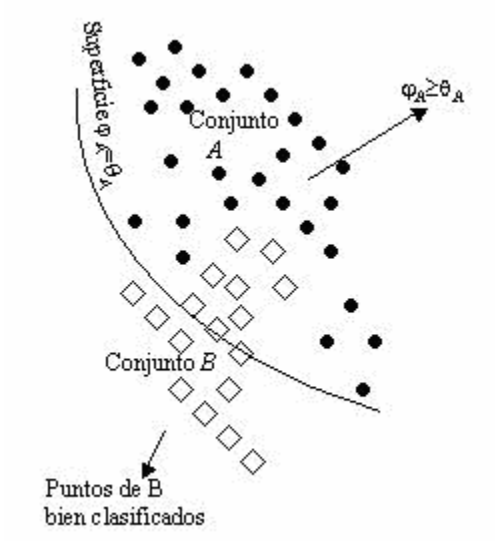

Figura 15: Hiper-superficie contentiva de los puntos del conjunto A

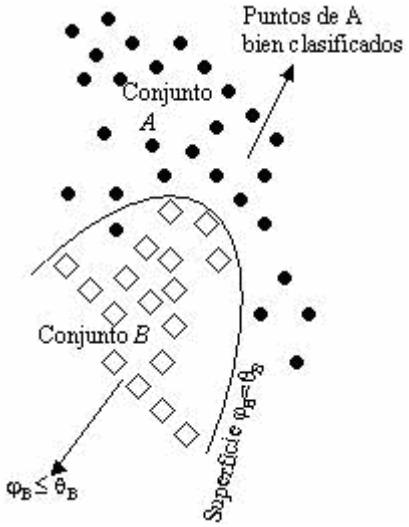

Figura 16: Hiper-superficie contentiva de los puntos del conjunto B

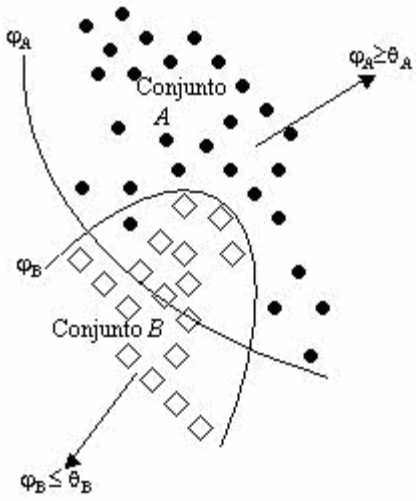

Figura 17: Par de Hiper-superficies

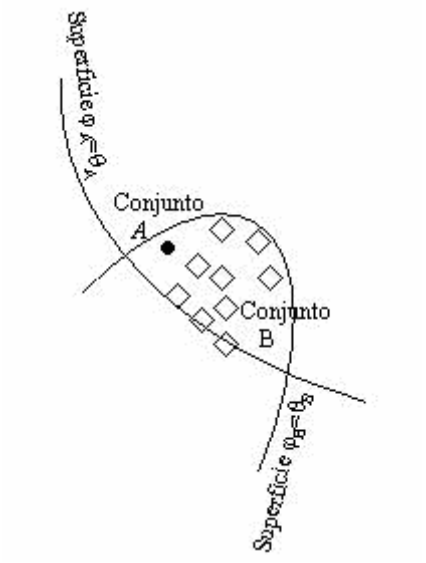

Figura 18: Puntos sin clasificar por Hiper-superficies

3.3 Nomenclatura de los Métodos Multi-superficie

Es opinión del autor que la nomenclatura de este método general debe indicar que es un método “multi-superficies” (tal y como llama Mangasarian a su método en [Man92]), pero dado que ese nombre está siendo utilizado para referirse al método de Mangasarian, es necesario realizar una nueva serie de definiciones, incluyendo una nueva forma de nombrar al método MSM de Mangasarian.

Una forma razonable de bautizar el método general expuesto acá es llamarlo “Optimización Generativa de Multi-Superficies” (OGMS). Si se acepta esta nomenclatura para el método más general, entonces se puede ahora hacer referencia al propuesto por Mangasarian renombrando a éste como “Optimización Generativa de Multi-Superficies Lineales Generadas a Pares Paralelos” (OGMS-

LGPP). Nótese que se han agregado términos indicando la linealidad, paralelismo y simultaneidad de las superficies generadas por iteración. Es cierto que puede definirse más de un método que cumpla con lo descrito en esa expresión, aparte del de Mangasarian. Sin embargo, no hay ningún otro método con esas características, que sea del conocimiento del autor. Se decidió entonces proponer ese nombre para el método de Mangasarian ya que, siguiendo el principio de la Navaja de Ockham, es la opción más sencilla.

Podrían hacerse cambios en la nomenclatura que proporcionen mayor información acerca de los criterios de minimización de errores, tales como el uso de las Normas 1, 2 e infinito, o también la minimización de los parámetros de las superficies adicionalmente a los errores. Sin embargo por los momentos la nomenclatura propuesta parece suficiente. A medida que surjan más métodos de este tipo, con seguridad se hará necesario aumentar la especificidad de la nomenclatura.

3.4 Instancia del Método General Propuesto

Para poder evaluar la bondad del método general propuesto, resulta necesario compararlo con uno que cumpla con el mismo objetivo, por ejemplo el OGMS-LGPP (antes nombrado simplemente MSM o “Método Multi-superficies”) de Mangasarian [Man92]. Con la finalidad de hacerlo comparable, se presenta una formulación que constituye, en si misma, una propuesta adicional, que recibe el nombre de “Optimización Generativa de Multi-Superficies Lineales Generadas a Pares” (OGMS-LGP).

Para este caso particular se ha formulado el siguiente modelo de Programación Lineal para obtener las superficies $\varphi_A(x) = \theta_A$ y $\varphi_B(x) = \theta_B$ minimizando los errores de

clasificación de los elementos del conjunto contrario al que se desea abarcar por completo:

a) Para conseguir los parámetros de la Superficie $\varphi_A(x) = \theta_A$:

$$\begin{aligned}
&\min_{w_A,\theta_A,y_B} \langle e, y_B \rangle \\
&sujeto\ a: \\
&\quad Aw_A \geq e(\theta_A + 1) \\
&\quad Bw_A - y_B \leq e(\theta_A - 1) \\
&\quad y_B \geq 0
\end{aligned}$$

Modelo 3: PL que minimiza los elementos de $\mathcal{B}$ mal clasificados

b) Para conseguir los parámetros de la Superficie $\varphi_B(x) = \theta_B$:

$$\begin{aligned}
&\min_{w_B,\theta_B,y_A} \langle e, y_A \rangle \\
&sujeto\ a: \\
&\quad Aw_B + y_A \geq e(\theta_B + 1) \\
&\quad Bw_B \leq e(\theta_B - 1) \\
&\quad y_A \geq 0
\end{aligned}$$

Modelo 4: PL que minimiza los elementos de $\mathcal{A}$ mal clasificados

Como puede observarse, tanto el Modelo 3 como el Modelo 4 corresponden a uno de los objetivos perseguidos, y constituyen una pequeña *pero significativa* variación del modelo PL utilizado por el algoritmo de Mangasarian, basado en el PL robusto de Bennett y Mangasarian [BM92a]. Sin embargo, dada la sencillez de los modelos, bien puede obtenerse ambas superficies con un modelo ampliado, como se muestra a continuación en el Modelo 5:

$$\min_{w_A, w_B, \theta_A, \theta_B, y_A, y_B} \left(\frac{\langle e, y_A \rangle}{m} + \frac{\langle e, y_B \rangle}{k} \right)$$

$$\begin{aligned} & sujeto\ a: \\ & \quad Aw_A \geq e(\theta_A + 1) \\ & \quad Bw_A - y_B \leq e(\theta_A - 1) \\ & \quad Aw_B + y_A \geq e(\theta_B + 1) \\ & \quad Bw_B \leq e(\theta_B - 1) \\ & \quad y_A, y_B \geq 0 \end{aligned}$$

Modelo 5: PL ampliado para obtener 2 hiper-planos no-paralelos

Los parámetros de las superficies que se consiguen al resolver este PL “ampliado” son exactamente los mismos que los que se pueden conseguir al resolver Modelo 3 y Modelo 4. En otras palabras, resolver el Modelo 5 es equivalente a resolver el Modelo 3 y el Modelo 4 por separado.

Si la cantidad de puntos es muy grande puede ser contraproducente resolver el sistema ampliado (Modelo 5), por lo que se pueden resolver por separado Modelo 3 y Modelo 4, teniendo cada uno la mitad de las variables de decisión y la mitad de las restricciones. Si el tamaño del problema no es excesivamente grande, puede utilizarse el Modelo 5.

3.4.1 Comparación con el MSM

Para tener una idea más clara de cómo se relaciona este modelo PL con el planteado en cada una de las iteraciones del MSM propuesto por Mangasarian [Man92], puede verse que este último, al ser resuelto, y generarse los dos planos paralelos construidos en base al “mejor hiper-plano” de separación, realiza un trabajo *semejante* al que se logra resolviendo la siguiente formulación:

$$\min_{w_A, w_B, \theta_A, \theta_B, y_A, y_B} \left(\frac{\langle e, y_A \rangle}{m} + \frac{\langle e, y_B \rangle}{k} \right)$$

$$\begin{aligned} & sujeto\ a: \\ & \quad Aw_A \geq e(\theta_A + 1) \\ & \quad Bw_A - y_B \leq e(\theta_A - 1) \\ & \quad Aw_B + y_A \geq e(\theta_B + 1) \\ & \quad Bw_B \leq e(\theta_B - 1) \\ & \quad w_B = w_A \\ & \quad y_A, y_B \geq 0 \end{aligned}$$

Modelo 6: PL ampliado con restricción de paralelismo

Para aceptar la semejanza, hay que hacer la salvedad de que en este Modelo 6, el par de hiper-planos generados no son desplazados hasta “tocar” a alguno de los puntos de $\mathcal{A}$ o de $\mathcal{B}$, sino que yacen en un intermedio dentro del margen de separación. Sin embargo eso puede conseguirse también en el MSM de Mangasarian haciendo pequeñas modificaciones al algoritmo, por lo que esta diferencia no es necesariamente de relevancia.

Adicionalmente, en teoría la formulación en Modelo 6 no debería producir los resultados degenerados que se pueden apreciar en el algoritmo de Mangasarian. En otras palabras, el MSM de Mangasarian (OGMS-LGPP según la nomenclatura propuesta) es muy semejante al modelo de planos paralelos propuesto en el Modelo 6, aunque es inferior a este último en calidad de generalización y en propensión a resultados degenerados. Sin embargo, el desliz que se comete al decir que el trabajo hecho en el MSM en cada iteración es semejante al Modelo 6 en realidad tendería a beneficiar al MSM en la comparación.

Nótese que el Modelo 6 es equivalente al Modelo 5, con la restricción adicional $w_A = w_B$. Con esa adición al modelo, las restricciones que incluyen a las variables de errores consiguen el mejor hiper-plano de separación ya que corresponden a las

restricciones del PL robusto [BM92a] que se resuelve en el MSM de Mangasarian [Man92], salvando la diferencia en el θ_A y el θ_B. Las otras dos restricciones, que carecen del término del error, simplemente realizan la traslación de los hiper-planos paralelos hasta llevarlos a un punto en el que separen elementos de un solo conjunto. La formulación incorpora la traslación de los dos hiper-planos, cosa que el algoritmo MSM de Mangasarian realiza luego de resuelto el PL robusto [Man92].

Así pues, queda demostrado que el trabajo logrado por el MSM de Mangasarian [Man92] es inferior, aunque semejante, al logrado por la formulación del Modelo 6. Si se recuerda que en esta última hay un grupo de restricciones adicional a las que posee la formulación del Modelo 5, se evidencia que, como se dijo en la sección 3.1, la instancia propuesta, el OGMS-LGP, garantiza una solución necesariamente mejor que la alcanzada por el método MSM de Mangasarian [Man92], en cada iteración.

3.4.2 Modelo de Optimización

Existen algunas convenciones que pueden utilizarse para facilitar la exposición de este tipo de modelos, que se ha tomado del trabajo realizado por González Castaño, García Palomares y Meyer en [GGM04]. La convención consiste en tomar $\sigma_A = \sigma(x_i | x_i \in \mathcal{A}) = 1$, y $\sigma_B = \sigma(x_i | x_i \in \mathcal{B}) = -1$, para que el modelo ampliado se pueda expresar de manera más organizada, como en los Modelos 7 y 8:

En este caso, la superficie $\varphi_A(x)=\theta_A$ corresponde a la expresión $\langle w_A, x\rangle - 1 = \theta_A$. y $\varphi_B(x)=\theta_B$ corresponde a la expresión $\langle w_B, x\rangle - 1 = \theta_B$. En todos los casos se exige que los puntos de $\mathcal{A}$ estén del lado del semi-espacio definido por el “mayor que” de las

$$
\begin{aligned}
&\min_{w_A, w_B, \theta_A, \theta_B, y_A, y_B} \left(\frac{\langle e, y_A \rangle}{m} + \frac{\langle e, y_B \rangle}{k} \right) \\
&sujeto\ a: \\
&\quad \sigma_A A w_A \geq e(\sigma_A \theta_A + 1) \\
&\quad \sigma_A A w_B + y_A \geq e(\sigma_A \theta_B + 1) \\
&\quad \sigma_B B w_A + y_B \geq e(\sigma_B \theta_A + 1) \\
&\quad \sigma_B B w_B \geq e(\sigma_B \theta_B + 1) \\
&\quad y_A, y_B \geq 0
\end{aligned}
$$

Modelo 7: PL ampliado con indicador de signo

$$
\begin{aligned}
&\min_{w_A, w_B, \theta_A, \theta_B, y_A, y_B} \left(\frac{\langle e, y_A \rangle}{m} + \frac{\langle e, y_B \rangle}{k} \right) \\
&sujeto\ a: \\
&\quad \sigma_A (A w_A - e\theta_A) \geq e \\
&\quad \sigma_A (A w_B - e\theta_B) \geq e - y_A \\
&\quad \sigma_B (B w_A - e\theta_A) \geq e - y_B \\
&\quad \sigma_B (B w_B - e\theta_B) \geq e \\
&\quad y_A, y_B \geq 0
\end{aligned}
$$

Modelo 8: PL ampliado con indicador de signo factorizado

restricciones, y que los puntos de $\mathcal{B}$ estén del lado del semi-espacio definido por el “menor que” de las restricciones.

Esta forma de expresar el modelo permite observar las similitudes y diferencias estructurales entre las diferentes restricciones.

3.4.3 Comportamiento de las Superficies Generadas

En la Figura 19 se describe, geométricamente, un ejemplo de una superficie por partes que ilustra el comportamiento del algoritmo al usar la formulación propuesta.

Primero se construye, a partir de la formulación del OGMS-LGP un par de hiper-planos $\langle w^1{}_A, x\rangle = \theta^1{}_A$ y $\langle w^1{}_B, x\rangle = \theta^1{}_B$ tal que el semi-espacio $\{x | \langle x, w^1{}_B\rangle > \theta^1{}_B\}$ contiene puntos sólo de $\mathcal{A}$, mientras que el semi-espacio $\{x | \langle x, w^1{}_A\rangle < \theta^1{}_A\}$ contiene puntos sólo de $\mathcal{B}$. Los puntos separados por estos semi-espacios son descartados y el proceso se repite para los puntos que se encuentran entre esos hiper-planos, obteniéndose los hiper-planos $\langle w^2{}_A, x\rangle = \theta^2{}_A$ y $\langle w^2{}_B, x\rangle = \theta^2{}_B$, que, nuevamente, separan nuevos puntos de $\mathcal{A}$ y de $\mathcal{B}$. Descartados esos puntos ya clasificados nuevamente, los puntos restantes entre los planos $\langle w^2{}_A, x\rangle = \theta^2{}_A$ y $\langle w^2{}_B, x\rangle = \theta^2{}_B$ son separados por el hiper-plano $\langle w^3, x\rangle = \theta^3$, que puede tomarse tanto como el hiper-plano $\langle w^3{}_A, x\rangle = \theta^3{}_A$, como también por el hiper-plano $\langle w^3{}_B, x\rangle = \theta^3{}_B$, cuando en el

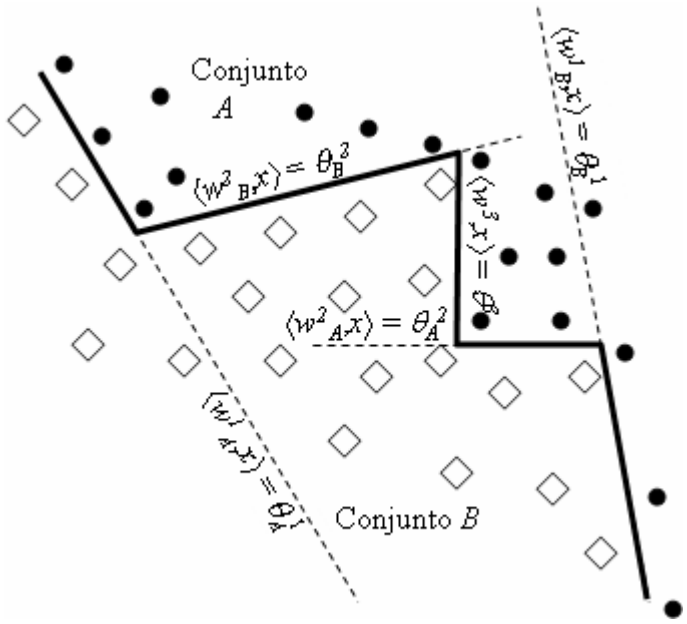

Figura 19: Tres iteraciones de la instancia propuesta

modelo el valor de la función objetivo es nulo. Esto puede hacerse porque la nulidad de la función objetivo, en este caso, significa que los puntos son linealmente separables por cualquiera de los hiper-planos hallados por el modelo formulado en esa iteración.

La separación lineal por partes de la Figura 19 es esquematizada como un árbol de decisión en la Figura 20.

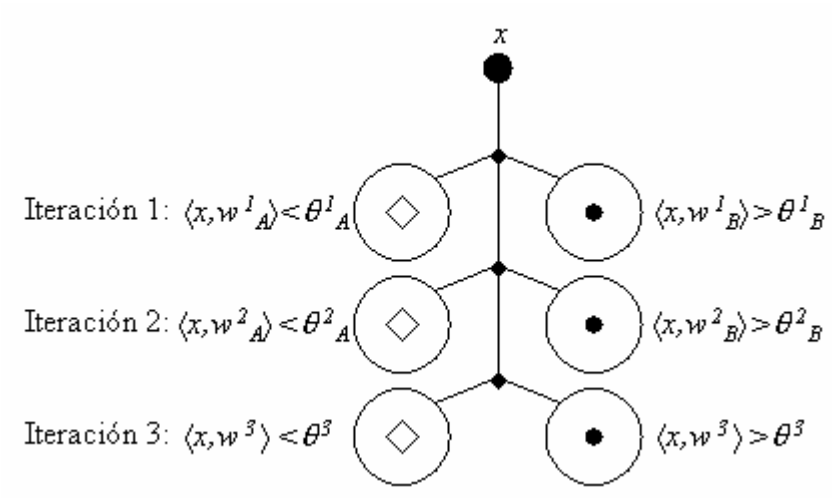

Figura 20: Árbol binario de la instancia generada

3.4.4 Algoritmo OGMS-LGP

Ahora se describe formalmente al OGMS-LGP. Nótese que se omite el procedimiento de anti-degeneración propuesto para el MSM por Mangasarian, que se usa cuando algunos pares de planos no separan punto alguno, tal que $\mathcal{A}^{i+1} = \mathcal{A}^{i}$ y $\mathcal{B}^{i+1} = \mathcal{B}^{i}$ para algún i en la terminología del algoritmo mostrado a continuación. Esto ocurre raramente, según Mangasarian [Man92]. Es de suponer que, al no exigir paralelismo

a los hiper-planos, los procedimientos de anti-degeneración son aún más raramente necesarios que en el MSM.

Sean $\mathcal{A}$ y $\mathcal{B}$ conjuntos de puntos disjuntos en R^n, representados por las matrices A y B, mxn y kxn, respectivamente. Denótese al Modelo 5 como $Opt(A,B)$, y $\arg\min[Opt(A,B)]$ a su solución. El procedimiento se describe en el Algoritmo 4.

Procedimiento: Optimización Generativa de Multi-Superficies Lineales Generadas a Pares

Comienzo

Paso 1: Inicialización

a) Inicialización del nivel del árbol (i:=0);
b) Inicialización del conjunto $\mathcal{A}$ de ese nivel ($\mathcal{A}^0$:=$\mathcal{A}$);
c) Inicialización del conjunto $\mathcal{B}$ de ese nivel ($\mathcal{B}^0$:=$\mathcal{B}$);

Paso 2: Procedimiento a ejecutar en el nivel i del árbol

Si ($\mathcal{A}^i \approx \emptyset$) **ó** ($\mathcal{B}^i \approx \emptyset$) **ó** (alguna condición que indique sobre-ajuste a los datos) **Entonces**

Parar

En Caso Contrario

a) Hallar los parámetros de las superficies de separación resolviendo $Opt(A^i, B^i)$.
$(w^i_A, w^i_B, \theta^i_A, \theta^i_B, y^i_A, y^i_B) \in arg\ min\ (Opt(A^i, B^i))$
b) Definir los descendientes de $\mathcal{A}^i$ y $\mathcal{B}^i$:
A^{i+1}:={ $A^i_m \mid A^i_m w^i_B \leq \theta^i_B$}, B^{i+1}:={ $B^i_m \mid B^i_m w^i_A \geq \theta^i_A$}

Fin de Condicional

Paso 3: Actualizaciones para próxima iteración/nivel del árbol

a) Actualización del nivel del árbol (i:=i+1);
b) **Ir a** Paso 2

Fin

Algoritmo 4: OGMS-LGP

3.4.5 RNA Generada

Ha de notarse que la separación binaria que ha realizado el algoritmo OGMS-LGP es equivalente a una red de perceptrones de una sola capa oculta [BM92b] tal y como se muestra más adelante, en la Figura 21. La misma se ha elaborado en concordancia con el modelo mostrado en la Figura 19 que a su vez corresponde al árbol de decisión de la Figura 20. A continuación se describe detalladamente el funcionamiento de una RNA construída y entrenada de esa forma.

El primer par de perceptrones de la capa escondida, con sus arcos de entrada, representa a los hiper-planos $\langle w^1{}_A, x\rangle = \theta^1{}_A$ y $\langle w^1{}_B, x\rangle = \theta^1{}_B$. Tomando la convención de que $s(x)$=+1 si $x \in \mathcal{A}$, y $s(x)$=-1 si $x \in \mathcal{B}$, se puede ver que la red clasifica correctamente los puntos. Para verificar esto, nótese que si un x cumple con que $\langle w^1{}_B, x\rangle > \theta^1{}_B$ entonces el primer perceptrón dispara un valor de s=+1, y con toda seguridad el segundo también lo hará. Nótese que en la formulación del problema de optimización no puede haber puntos que cumplan al mismo tiempo $\langle w^1{}_B, x\rangle > \theta^1{}_B$ y $\langle w^1{}_A, x\rangle < \theta^1{}_A$. Por esto, ambas neuronas sumarán un total de 8 en la entrada de la neurona de salida, provocando una salida de la red igual a +1. Si, por el contrario, un x cumple con $\langle w^1{}_A, x\rangle < \theta^1{}_A$, la segunda neurona disparará un valor de –1, y también lo hará la primera neurona, para sumar, en la entrada de la neurona de salida, un total de –8, que es evidentemente menor que cero, ocasionando una respuesta final de la red igual a –1.

Los pesos que presenta la neurona de la última capa, en el tipo de red que se genera al aplicar este método, no son arbitrarios. La forma en que se asignan ha sido diseñada en esta heurística, de tal manera que la red se comporte como un árbol de decisión del tipo mostrado en la Figura 20. Si se consideran los pesos y

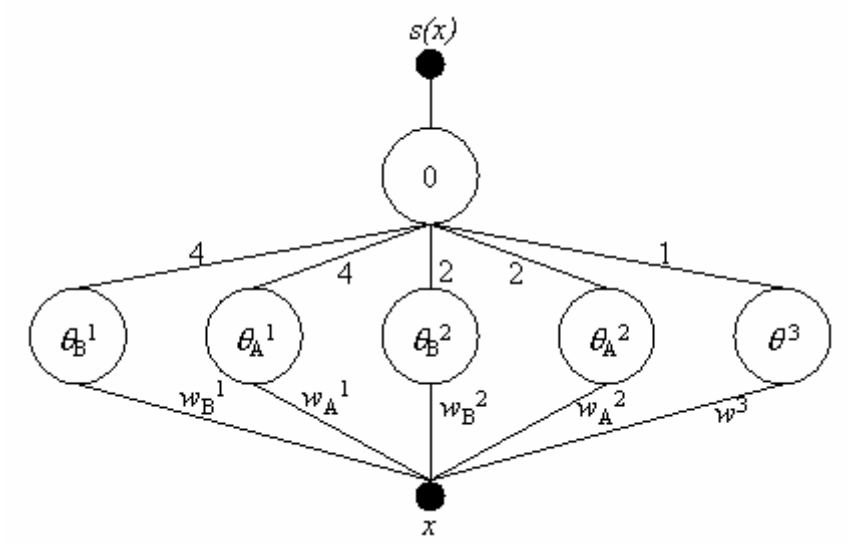

Figura 21: RNA generada por el OGMS-LGP

umbrales de las neuronas de la capa oculta, con el mismo signo que toman en el modelo de optimización utilizado para su hallazgo, los pesos de la neurona de salida deben poseer una característica muy específica, que puede lograrse de muchas maneras. Esta consiste en que los pesos del par de neuronas de la iteración i-ésima debe ser el mismo para cada una de ellas. Asimismo, el valor que tomen ambos pesos debe ser estrictamente mayor que la suma de los valores de los pesos que tiene la neurona de salida para las neuronas generadas en las iteraciones posteriores.

En el caso de la RNA de la Figura 21, cuando un punto ha sido clasificado correctamente por las neuronas de la primera iteración (la primera y segunda de izquierda a derecha), ambas neuronas arrojarán un valor idéntico de –1 o +1, dependiendo del conjunto al que pertenezca el punto, y lo que los otros perceptrones disparen no afecta a la respuesta de la red porque la ponderación hace que la respuesta de la red dependa únicamente de lo que disparen esas dos neuronas. Es decir, siempre que ambas neuronas disparen el mismo impulso, gracias a que $8>2+2+1$ serán las neuronas que dictaminarán la respuesta total de la red. Cuando x

está entre los hiper-planos $\langle w^1{}_A, x \rangle = \theta^1{}_A$ y $\langle w^1{}_B, x \rangle = \theta^1{}_B$, la salida de la primera neurona es de -1, que multiplicando por su peso (+4), contribuye con un valor de (-4) en la suma a la entrada a la neurona final. La segunda neurona arrojará un valor de +1, que, multiplicado por su peso (+4), compensará el efecto de la primera neurona. La contribución neta de ambas neuronas resulta nula (indicando que el punto aún está por clasificar), dejando la decisión en manos de los perceptrones de iteraciones posteriores, en este caso el 3º y el 4º. Ellos determinarán con una lógica similar, si $x \in \mathcal{A}$ o si $x \in \mathcal{B}$. Si se anulan entre si los disparos de estas últimas, la decisión es tomada por la quinta y última neurona, que dispara +1 si $x \in \mathcal{A}$, y -1 si $x \in \mathcal{B}$.

Resulta evidente que la estrategia de asignación de pesos para la entrada de la neurona de la capa de salida no es la única, tal y como se ha dicho antes. Estos valores pueden definirse de infinitas formas, siempre y cuando cumplan con la condición de que el efecto sumado de las neuronas de la i-ésima iteración sea mayor a la suma de los efectos de las neuronas i+1, i+2,... hasta la última iteración realizada. Si se escogen valores enteros como en el ejemplo planteado en las figuras mostradas arriba, los pesos han de irse desplazando de tal manera que el nuevo peso de mayor tamaño (siendo el más pequeño igual a la unidad), siempre esté en el primer par de neuronas generadas. Esto no sería necesario si, por ejemplo, se escogiesen valores fraccionales tales que la primeras neuronas generadas tuviesen un peso igual a la unidad, y los siguientes pesos siguiesen una serie de números fraccionales cuya suma cumpla con lo requerido. Tómese como ejemplo una serie P de pesos como la siguiente: $P = \left\{1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \ldots, \frac{1}{2^k}, \ldots\right\} \forall k = 0, 1, 2, \ldots$, donde evidentemente el término para k=0 sería el peso que se le asignaría a la salida de las neuronas de la primera iteración, y el término k-ésimo correspondería al del par de neuronas de la iteración (i+1).

3.4.6 Casos de Hiper-Planos Degenerados

Para cubrir los casos degenerados que pueden ocurrir (aunque ocurran poco, aún menos que en el OGMS-LGPP de Mangasarian), debe preverse la posibilidad de que: o $\mathcal{A}^{i} = \mathcal{A}^{i+1}$, o $\mathcal{B}^{i} = \mathcal{B}^{i+1}$. Ambas cosas no deberían ocurrir juntas porque las superficies que permiten descartar puntos de $\mathcal{A}$, y puntos de $\mathcal{B}$, son independientes en sus parámetros, pero esto aún es objeto de estudio por parte del autor. Sin embargo, intuitivamente resulta imposible para el autor (hasta el momento) pensar en una disposición de puntos diferentes tales que el modelo sea incapaz de encontrar al menos un punto de alguno de los conjuntos (siempre y cuando estos sean disjuntos), que se pueda descartar.

Cuando ocurra que una superficie sea creada por el algoritmo y no permita clasificar correctamente ningún punto (o una cantidad muy poco significativa de puntos, bajo algún criterio), podría pensarse que la neurona que está representando ese hiper-plano no está contribuyendo en la clasificación, y por lo tanto estaría sobrando, en la arquitectura de la red neuronal. Sin embargo, un hiper-plano con esas características, es necesario, tal y como se muestra en la Figura 22. Todos los puntos que se encuentran entre ese hiper-plano degenerado y el otro hiper-plano creado en esa iteración originan, en ese par de neuronas, una respuesta neta igual a cero (0), indicando que no se pueden clasificar todavía y hace falta una nueva iteración. Esto sucede gracias a que ese hiper-plano degenerado, pese a no clasificar ningún punto, sirve para anular con su disparo, el que realiza el otro hiper-plano al recibir como entrada esos puntos en territorio intermedio.

De eliminarse el hiper-plano degenerado, los puntos ubicados entre ambos hiper-planos pasarían a tener una clasificación de –1 o +1, cuando en realidad aún deberían ser estudiados en iteraciones posteriores. Recuérdese que al eliminarse hipotéticamente el hiper-plano degenerado, el hiper-plano restante dividiría el R^n en

dos semi-espacios, uno al cual le asignaría la clasificación de +1 y otro al cual le asignaría la clasificación de –1.

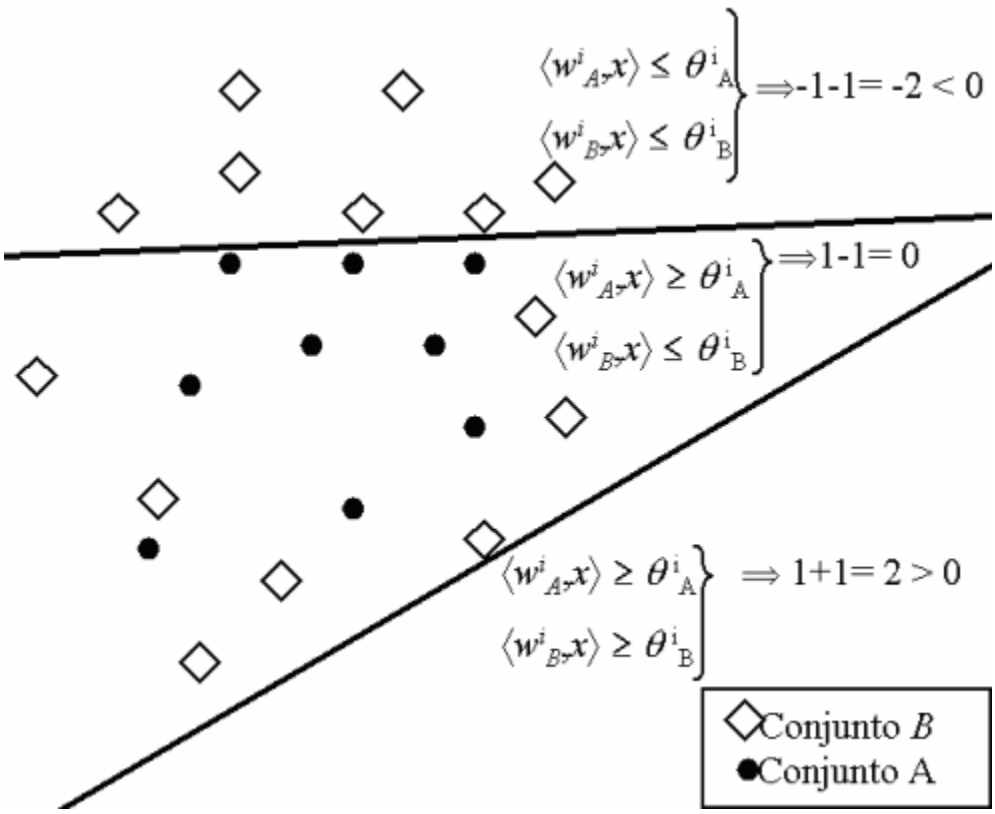

Figura 22: Ejemplo de iteración con un hiper-plano degenerado

3.4.7 Algoritmo de Eliminación de Hiper-Planos Degenerados

Gracias a esta propiedad los puntos en la zona intermedia de los hiper-planos resultan no clasificados, al tener un valor de clasificación igual a 0. Sin embargo, se propone una estrategia para proceder a la eliminación de las neuronas o hiper-planos degenerados sin asignar una clasificación falsa a los puntos que en principio no han sido clasificados aún. Esta estrategia consiste en cambiar la función de activación del perceptrón correspondiente al hiper-plano que queda des-apareado al eliminar el hiper-plano degenerado. De esta forma, si el mismo pretendía clasificar bien puntos

de $\mathcal{A}$, la salida correspondiente al disparo de inhibición se altere para ser 0 en lugar de –1. Mientras que si, por el contrario, el mismo pretendía clasificar bien puntos de $\mathcal{B}$, la salida correspondiente a +1 se cambie a –1.

En la Figura 22 se muestra lo explicado arriba. Nótese que la estructura de localización de los datos puede hacer imposible trazar un hiper-plano que logre separar puntos del conjunto $\mathcal{A}$, por cuanto están insertos dentro del enjambre de puntos del conjunto $\mathcal{B}$ que se encuentran a su alrededor, tal y como se muestra en el caso hipotético de la Figura 22.

En la figura los semi-espacios correspondientes a los “mayor que”, siempre están por “debajo” de las líneas que representan a cada hiper-plano. Puede verse que los puntos de $\mathcal{B}$ que pudieron ser clasificados correctamente por el hiper-plano $\langle w^i{}_A, x \rangle = \theta^i{}_A$, se encuentran en una región que pertenece al lado “menor que” de los dos hiper-planos considerados. Los puntos en la región intermedia de los dos hiper-planos, están del lado “mayor que” del hiper-plano $\langle w^i{}_A, x \rangle = \theta^i{}_A$, pero del lado “menor que” del hiper-plano $\langle w^i{}_B, x \rangle = \theta^i{}_B$, por lo que los disparos de las neuronas correspondientes a dichas superficies se anulan entre si dando como resultado una clasificación igual a cero (0) (por lo que el algoritmo debe continuar generando hiper-planos). Finalmente, la tercera y última región se encuentra “debajo” del hiper-plano $\langle w^i{}_B, x \rangle = \theta^i{}_B$, e idealmente pretende contener a los puntos que logran clasificarse satisfactoriamente, pertenecientes al conjunto $\mathcal{A}$. Por esa razón, cualquier punto en esa región del espacio R^n habría satisfecho los lados “mayor que” de los hiper-planos considerados en el ejemplo. Sin embargo, dada la configuración de los puntos allí dibujados, es imposible aislar punto alguno del conjunto $\mathcal{A}$, utilizando una superficie de ese tipo, por lo que el hiper-plano resulta degenerado, y su función, como se explicó anteriormente, se limita a permitir que los que se encuentren en la región

intermedia, no sean considerados como clasificados. El único criterio que define la posición de ese hiper-plano, es que intenta minimizar las distancias que tienen los puntos del conjunto $\mathcal{A}$ respecto a él (bajo alguna definición de norma que se considere apropiada).

Como se acaba de explicar, el método propuesto para solucionar el problema de los hiper-planos degenerados mediante su eliminación, consiste en obviar la existencia del hiper-plano degenerado, y cambiar la función de transferencia del perceptrón restante. En el caso del hiper-plano que queda luego de la eliminación, en el ejemplo dado, puede verse en la Figura 23.

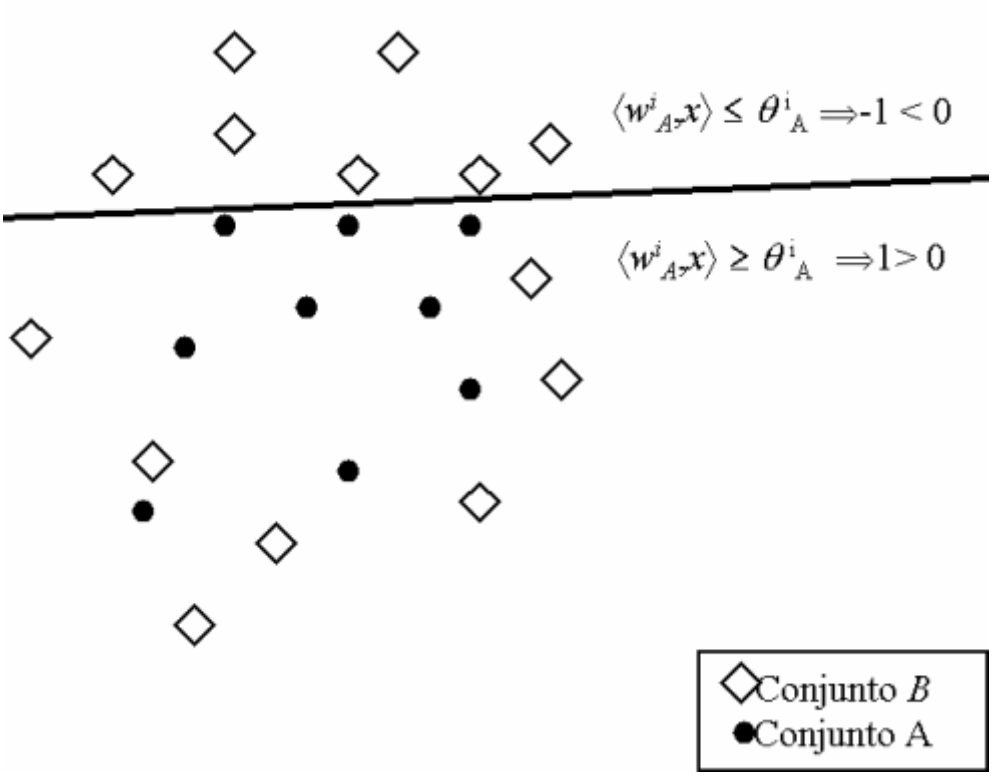

Figura 23: Eliminación de hiper-plano degenerado sin cambiar función de transferencia

En este caso interesa que la función de transferencia, al recibir una entrada $z<0$, arroje un resultado de –1 (que corresponde con el código asignado al conjunto $\mathcal{B}$), y

al recibir una entrada $z>0$ (como sucede en la zona inferior de la figura), el disparo sea cero (0). Esto se muestra en la Figura 24.

Existen dos formas alternativas de eliminar los hiper-planos degenerados:

a) Eliminar el hiper-plano durante la iteración y realizar los cambios en la función de transferencia al hiper-plano restante “en vivo” (al ejecutarse esa iteración).
b) Marcar los hiper-planos potencialmente descartables y eliminarlos al final de la totalidad de las iteraciones.

Se sospecha que el marcaje de los hiper-planos potencialmente descartables representa un consumo innecesario de memoria, por lo que se prefiere la primera de las opciones, quedando el procedimiento como se muestra en el Algoritmo 5.

La posible eliminación de *hasta* un hiper-plano por iteración hace surgir la siguiente pregunta: *¿por qué generar dos superficies de clasificación en vez de una, en cada iteración?* En este momento el autor está investigando para responder esa interrogante. Para ello se están planificando ensayos con RNAs que utilicen, desde un principio, neuronas de salida +1/0 y 0/-1. Sin embargo, tales propuestas, así como las estrategias para implementación física de las RNAs que cumplan con estas características, no serán desarrolladas aquí. Esto es porque, aunque forman parte de los planes de investigación para el futuro, escapan al alcance definido para el presente Trabajo de Grado.

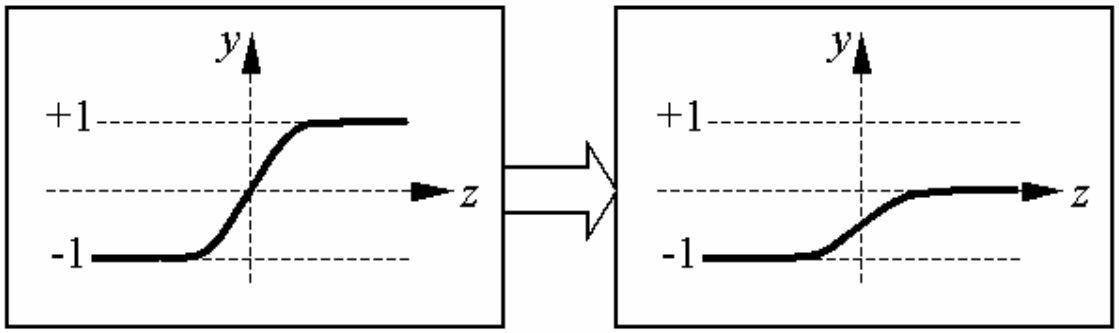

Figura 24: Cambio en función de transferencia para eliminar hiper-plano degenerado

Procedimiento para Degeneraciones para el Algoritmo MSM

Para insertarse antes del paso 3 del algoritmo OGMS-LGP (Algoritmo 4)

Comienzo

Paso 2.2: Eliminar Hiper-planos Degenerados

Si ($\mathcal{A}^{i} = \mathcal{A}^{i+1}$) **Entonces**

a) **Eliminar** la neurona-perceptrón correspondiente al hiper-plano $\langle w^{i}_{B}, x\rangle = \theta^{i}_{B}$

b) **Definir** como igual a 0 el nivel superior de la función de activación de la neurona correspondiente al hiper-plano $\langle w^{i}_{A}, x\rangle = \theta^{i}_{A}$

Fin de Condicional

Si ($\mathcal{B}^{i} = \mathcal{B}^{i+1}$) **Entonces**

a) **Eliminar** la neurona-perceptrón correspondiente al hiper-plano $\langle w^{i}_{A}, x\rangle = \theta^{i}_{A}$

b) **Definir** como igual a 0 el nivel superior de la función de activación de la neurona correspondiente al hiper-plano $\langle w^{i}_{B}, x\rangle = \theta^{i}_{B}$

Fin de Condicional

Paso 2.3: continuar con el OGMS-LGP en el Paso 3

Fin

Algoritmo 5: Eliminación de hiper-planos Degenerados en OGMS-LGP

3.5 Sobre los criterios de optimalidad

Dependiendo de lo que se considere el “mejor” hiper-plano, o lo que es lo mismo, el “mejor” perceptrón, se pueden considerar diversos métodos y criterios de optimización. Se ha dicho en la introducción a este trabajo que es imposible minimizar el riesgo real de clasificación, aunque es perfectamente posible minimizar el riesgo empírico. En el caso de este algoritmo, el riesgo empírico está asociado a la

magnitud del error que se mide en la función objetivo del modelo de optimización utilizado para generar los pares de superficies. Este puede ser minimizado en las sucesivas iteraciones tanto como para que el error de clasificación de toda la red sea exactamente cero. Sin embargo, tal y como se explicó antes, esto no necesariamente conduce a obtener una red que minimice el riesgo real.

El riesgo empírico, contabilizado en función de la cantidad de errores que se cometen en la clasificación de los elementos del conjunto de entrenamiento, en la mayoría de los casos en los que se llega a eliminar por completo, produce una red que pierde la generalidad necesaria para clasificar los puntos que en un futuro se le presenten a la red para su clasificación. En el procedimiento experimental se utilizó una estrategia de *10-Fold Validation* (Validación por 10 Grupos) para medir la certeza de clasificación en conjuntos de validación, lo que permitirá estimar en qué iteración la red empieza a perder generalidad (sin importar cuáles datos sean escogidos para el entrenamiento), y, por lo tanto, permitirá estimar el “mejor” tamaño de la RNA para el problema propuesto. Sin embargo, no hay definición matemática de lo que es la “mejor” red. Sólo se sabe que se desea obtener una red neural, con las características que han sido discutidas a lo largo de este trabajo, que generalice “suficientemente bien” como para que tenga sentido su uso en la práctica. No existe criterio único para cumplir con esto, y con frecuencia sólo la experiencia puede fundamentar la valoración de cada criterio.

Existen métodos constructivos de diseño de RNAs de re-alimentación positiva que utilizan criterios que van desde la clásica minimización del error cuadrático medio (o, equivalentemente, la Norma-2 del error cometido), hasta la minimización de ciertas expresiones matemáticas asociadas al concepto de entropía [AG97].

En particular se han considerado, en la presente investigación, los siguientes criterios para ser incluidos en los modelos de optimización utilizados en las iteraciones del método propuesto:

a) Minimización de Errores en Norma-1:

En el algoritmo OGMS-LGP (expuesto en la sección 3.4) se tomó como criterio de minimización de errores la Norma-1, debido a que ese tipo de problemas de optimización (Programación Lineal) es uno de los métodos más exitosos en investigación de operaciones por poseer una respuesta polinomial que le permite la solución de problemas de gran tamaño; además, hace que nuestra formulación sea comparable con el algoritmo MSM de Mangasarian [Man92], lo cual es de interés para mostrar la relevancia de las estrategias que constituyen el principal aporte de la presente investigación. La forma que toma este criterio en el OGMS-LGP es $\min\langle e, y\rangle$ donde y es la Norma-1 del error.

La Norma-1 se puede entender, de forma intuitiva, como la minimización de los errores promedio de las distancias que hay entre los puntos mal clasificados hasta la superficie que los debería clasificar correctamente. Sin embargo en este caso la palabra “distancia” no se refiere a la distancia euclidiana, sino al valor de las variables de error, que es proporcional a la “verdadera distancia” que hay desde el punto hasta su proyección en superficie de separación. La proporcionalidad entre el valor de esas variables y esa “distancia” depende de la magnitud del vector w asociado al hiper-plano. Cabe preguntar ¿Es esta valoración de los errores “mejor” que la Norma-2? Sólo los resultados prácticos y la satisfacción del usuario del modelo pueden dar una respuesta pertinente a esa pregunta.

b) Minimización de Errores en Norma-2:

Las propiedades de este criterio son explotadas de manera muy fructífera en modelos estadísticos de reconocimiento de patrones que utilizan premisas de índole probabilístico. Sin embargo, a juicio del autor, resulta poco práctico para ser implementado en el OGMS-LGP. Esto se debe a que se necesita realizar el

trabajo de elevar las distancias al cuadrado y resolver el problema como cuadrático, pudiendo resolverse de manera lineal. Sin embargo, el autor reconoce que dicho criterio cuenta con la ventaja de tener una interpretación geométrica fácil de entender y que resulta, en principio, la más intuitiva de las formas de conceptualizar las “distancias” entre puntos. La “verdadera distancia” entre los puntos y cada hiper-plano de separación está asociada a la norma del vector w.

c) Minimización de Errores en Norma-∞:

La Norma-∞, en este contexto, consiste en el valor absoluto del mayor de los errores. Resulta muy poco intuitivo este concepto de magnitud del vector de errores, y sin embargo, en algunos problemas específicos puede resultar de mucha utilidad. La minimización del máximo componente del vector de errores puede tener sentido cuando el promedio de errores sea un indicador poco relevante. Un ejemplo útil para comprender en qué tipo de contextos puede ser útil modelar, utilizando este criterio, es el caso de ubicación de centrales de ambulancias. Uno no desea que en promedio las ambulancias lleguen a tiempo para evitar que los pacientes de la zona se mueran. Por el contrario, se desea minimizar el peor de los tiempos de llegada al sitio, para que aún en ese caso, el paciente tenga una esperanza de vida razonable.

El autor no consideró pertinente utilizar este criterio dado que en la mayor cantidad de aplicaciones el error promedio es un indicador relevante. Ha de señalarse que existen formas de implementar este criterio de tal manera que el problema sea de Programación Lineal.

d) Minimización de los vectores de pesos:

El rol de la Norma-2 del vector de los pesos de un perceptrón en la función objetivo, es fundamental para que un modelo de optimización que esté tratando

de realizar una clasificación binaria pueda trabajar. Este es un principio bien conocido en el campo de estudio de los Support Vector Machines [ASS00].

En general, la magnitud del vector de pesos está asociada con la capacidad de generalización que posee la superficie de clasificación, en cuanto a que tiende a evitar que la superficie se acerque “demasiado” a los puntos que intenta clasificar. Generalmente se minimiza la Norma-2 del vector, ya que como se comentó antes, la misma tiene una interpretación geométrica muy evidente, y además de extrema utilidad cuando se utilizan pares de planos paralelos, como en el método de Programación Lineal Robusta de Mangasarian y Bennett [BM92a].

A juicio del autor, aunque la minimización de la Norma-1 de este vector no presenta la interpretación geométrica de la que goza la Norma-2, numéricamente el efecto resultante es, de igual forma, la reducción de la magnitud del vector de pesos. Por ende, es lógico pensar que la minimización en Norma-1 (cuyas ventajas computacionales han sido mencionadas ya) también tenga el efecto deseado, que es el de aumentar la capacidad de generalización de las superficies de separación. No obstante, este criterio no fue incluido en la función objetivo del OGMS-LGP, debido a que el hacerlo hubiese hecho del algoritmo, uno menos comparable con el MSM de Mangasarian.

Como último comentario acerca de este criterio, puede decirse que una formulación que podría resultar práctica en su implementación es la mostrada en el Modelo 9. En este se realizó el cambio de variables $z_A = w_A + e\xi_A$ para los pesos de la superficie que separa elementos de $\mathcal{B}$ puros, y $z_B = w_B + e\xi_B$ para la superficie que separa elementos de $\mathcal{A}$ puros.

$$\min_{w_A,w_B,\theta_A,\theta_B,y_A,y_B,z_A,z_B,\xi_A,\xi_B}\left(\frac{\langle e,y_A\rangle}{m}+\frac{\langle e,y_B\rangle}{k}\right)+\lambda(\xi_A+\xi_B)$$

$$\begin{aligned}
&sujeto\ a:\\
&\quad A(z_A-e\xi_A) \geq e(\theta_A+1)\\
&\quad B(z_A-e\xi_A)-y_B \leq e(\theta_A-1)\\
&\quad A(z_B-e\xi_B)+y_A \geq e(\theta_B+1)\\
&\quad B(z_B-e\xi_B) \leq e(\theta_B-1)\\
&\quad y_A,y_B,z_A,z_B,\xi_A,\xi_B \geq 0
\end{aligned}$$

Modelo 9: PL ampliado con minimización indirecta de pesos

Esta formulación permite realizar una minimización de una *cierta* norma *w* conservando la linealidad del modelo. El costo en aumento de variables que ocasiona esta formulación es incluso menor que el que se tendría si se minimizara la Norma-1 del vector los vectores de pesos.

En este apartado del presente trabajo de investigación se ha comentado acerca de los criterios considerados como opciones interesantes para ser aplicadas dentro del método propuesto OGMS-LGP. Este fue diseñado con la intención de ofrecer un algoritmo comparable con el MSM de Mangasarian, y solamente eso. Este no pretende ser, por lo tanto, un estudio exhaustivo de las formas en que se puede implementar el método OGMS. Se ha decidido plasmar estos comentarios, sin embargo, para ofrecer al lector algo de orientación en cuanto a las bondades y defectos encontrados en los diferentes criterios de optimalidad.

3.6 Convergencia del OGMS-LGP

Considérese una red con *n* neuronas en la capa oculta (siendo *n* un número par igual al doble de las iteraciones, es decir, $n=2i$, donde i es el número de la iteración). Sea

ε^{i}_{A} el número de errores de clasificación de elementos del conjunto $\mathcal{A}$ en la iteración i-ésima, y ε^{i}_{B} el número de errores de clasificación de elementos del conjunto $\mathcal{B}$ en la misma iteración. Si ε^{i} contabiliza el número total de errores para la iteración i, entonces se tiene que $\varepsilon^{i} = \varepsilon^{i}_{A} + \varepsilon^{i}_{B}$.

Supóngase que las neuronas (n+1)-ésima y (n+2)-ésima (creadas en la (i+1)-ésima iteración) son utilizadas para seguir tratando de clasificar los puntos que han dejado por fuera las n neuronas anteriores. Se puede verificar que:

a) El método garantiza que una de las superficies generadas por estas dos neuronas clasifique elementos de $\mathcal{B}$, y la otra, elementos de $\mathcal{A}$.

b) Tomando superficies lineales (hiper-planos), que son la clase más sencilla de superficies, se puede garantizar que al menos un punto de toda la información de entrenamiento puede ser separado del resto por una superficie de ese tipo. Dicho de otra manera, al menos un punto de la información de entrenamiento es linealmente separable del resto. Esta es una común, intuitiva y demostrada propiedad de los puntos extremos de los cascarones convexos de conjuntos de puntos disjuntos.

Tomando en cuenta estos hechos, se puede afirmar que, sin importar si el punto o dato linealmente separable es de $\mathcal{A}$ o de $\mathcal{B}$, alguna de las dos superficies que se generan podrá clasificar bien ese dato. Si se consideran superficies menos rígidas y simples que los hiper-planos (cualquier curva no lineal), las condiciones de separabilidad son aún mejores, por lo cual es suficiente hacer el análisis de convergencia considerando el uso de hiper-planos como superficies de separación. Dicho de otra manera, el uso de superficies más complejas daría iguales o mejores resultados. Si la convergencia se puede probar para hiper-planos, queda entonces garantizada para cualquier curva más compleja.

Dicho esto, se puede demostrar que el algoritmo es capaz de clasificar, sin errores, cualquier cantidad finita de datos pertenecientes a dos conjuntos disjuntos, en una cantidad finita de iteraciones. Para ello es suficiente mostrar que cada par de neuronas generado en una iteración i hace decrecer el número de errores de clasificación *al menos* por un elemento (sea este del conjunto que fuere).

Continuando con la prueba, de lo antes mencionado se evidencia que ocurre uno de estos tres escenarios:

I.- $[\varepsilon^{i}{}_{A} - \varepsilon^{i+1}{}_{A} = 0]$, $[\varepsilon^{i}{}_{B} - \varepsilon^{i+1}{}_{B} \geq 1]$ (clasifica uno o más puntos de $\mathcal{B}$ y ninguno de $\mathcal{A}$)

II.- $[\varepsilon^{i}{}_{A} - \varepsilon^{i+1}{}_{A} \geq 1]$, $[\varepsilon^{i}{}_{B} - \varepsilon^{i+1}{}_{B} = 0]$ (clasifica uno o más puntos de $\mathcal{A}$ y ninguno de $\mathcal{B}$)

III.- $[\varepsilon^{i}{}_{A} - \varepsilon^{i+1}{}_{A} \geq 1]$, $[\varepsilon^{i}{}_{B} - \varepsilon^{i+1}{}_{B} \geq 1]$ (clasifica uno o más puntos *de ambos conjuntos*)

por lo tanto, aún en los peores casos (escenarios I y II mencionadas arriba) se cumple que $\varepsilon^{i+1}{}_{A} + \varepsilon^{i+1}{}_{B} \leq \varepsilon^{i}{}_{A} + \varepsilon^{i}{}_{B} - 1$, y como $\varepsilon^{i+1} = \varepsilon^{i+1}{}_{A} + \varepsilon^{i+1}{}_{B}$, podemos concluir que $\varepsilon^{i+1} = \varepsilon^{i+1}{}_{A} + \varepsilon^{i+1}{}_{B} \leq \varepsilon^{i}{}_{A} + \varepsilon^{i}{}_{B} - 1 = \varepsilon^{i} - 1$. Es decir que, por transitividad, necesariamente:

$$\varepsilon^{i+1} \leq \varepsilon^{i} - 1$$

Ecuación 10: Conclusión de la prueba de convergencia del OGMS-LGP

lo cual converge a 0 en una cantidad de iteraciones a lo sumo igual a la cantidad de datos en el conjunto de prueba. Sin embargo, el espíritu de la propuesta realizada en el presente Trabajo de Grado, es que el algoritmo no necesariamente debe ser ejecutado hasta clasificar por completo a los elementos a clasificar, sino que por el contrario, es deseable que pueda detenerse en el momento en que se detecte, de alguna forma, que la RNA que se está construyendo, ha empezado a perder capacidad de generalización. Entre los planes de investigación está la realización de

un estudio para proponer un indicador cuantitativo que proporcione información “en vivo”, para detectar las pérdidas de generalización y detener de forma automática el algoritmo.

4. EXPERIMENTOS, RESULTADOS Y CONCLUSIONES

En esta sección se realiza la comparación empírica del desempeño del algoritmo MSM de Mangasarian con el propuesto como instancia del método general de multi-superficies sugerido en este trabajo. La capacidad de generalización de estos métodos será estimada a partir de una comparación empírica de su desempeño en dos problemas sintéticos de moderada dificultad; con una estructura de datos que originan complicaciones en la clasificación de hecho superiores a las conseguidas en casos reales reseñados en la literatura. Por ejemplo, una ligera variación del MSM de Mangasarian ha logrado en la Base de Datos de Cáncer de Seno de la Universidad de Wisconsin [MA92] (disponible en ftp://ftp.ics.uci.edu/pub/machine-learning-databases) precisiones de clasificación del orden del 98%, mientras que el MSM logra una precisión de clasificación del 83,5% con el más sencillo de los casos sintéticos creados (valor correspondiente al conjunto de entrenamiento. La precisión es menor en el conjunto de validación). Posteriormente se expone lo que se concluye de los resultados obtenidos.

4.1 Selección del Algoritmo de Referencia

Se escogió sólo al algoritmo MSM de Mangasarian para compararse con la instancia generada del método general propuesto.

Existen diversas razones por las cuales no se consideró pertinente comparar el desempeño de éste con otras técnicas, que se exponen a continuación:

a) Pese a que el algoritmo propuesto puede considerarse fundamentalmente un algoritmo para el entrenamiento y diseño de redes neurales de clasificación binaria, no es útil, para los efectos de este trabajo, comparar su desempeño con el de una red entrenada mediante métodos asociados a gradientes, tales como el de Reto-propagación (con o sin métrica variable), sea que se usen o no estrategias constructivas o incrementales para escoger el número óptimo de neuronas. La razón es que este algoritmo presenta, ante este tipo de métodos, las mismas ventajas que el MSM de Mangasarian, debidas al uso de técnicas no asociadas a gradientes. Las desventajas de estos métodos han sido brevemente expuestas en la sección 2.1.7.

b) Otros métodos heurísticos como los Algoritmos Genéticos, tampoco se consideran relevantes para ser comparados con la propuesta de este trabajo, ya que la cantidad de parámetros que deben ser arbitrariamente escogidos (índices de mutación, entre-cruzamiento, patrones de codificación, etc.) es significativamente mayor a los que deben seleccionarse en el método propuesto (sólo ha de escogerse el valor del margen ϕ). Adicionalmente, su convergencia sólo se garantiza en un número infinito de iteraciones, cosa que no ocurre con métodos como el MSM de Mangasarian y el aquí propuesto.

c) Entre los tipos de superficies que existen para la clasificación binaria, están también las no-lineales implícitas en el Kernel de las Máquinas de Vectores de Soporte (SVM por sus siglas en inglés) que determinan un discriminador lineal en un espacio de atributos aumentado. El problema resultante es lineal en los parámetros, pero la superficie de decisión que resulta es no-lineal. Hay dos motivos por los cuales tampoco es útil comparar el algoritmo propuesto con este tipo de clasificadores binarios:

- Cada iteración del método propuesto puede también ser realizada en un espacio aumentado, logrando de igual forma superficies no-lineales.

- Por esta razón, podría concluirse que en cada iteración del método propuesto es una SVM. Resultaría injusto comparar la precisión de clasificación de un método que genera una superficie no-lineal, con la de otro que genera una superficie no-lineal por partes.

- La esencia de la propuesta, es mantener las suposiciones lo más sencillas posibles, y hacer la menor cantidad de suposiciones. En realidad, para diseñar correctamente un Kernel en una SVM, hay que escoger el tipo de superficies no-lineales que se desea utilizar. Bien podría ocurrir que la superficie no-lineal fuese innecesariamente o insuficientemente compleja, para el problema que se intenta resolver. La propuesta de este trabajo tiene como sub-producto el nivel de complejidad de la superficie por partes que se produce (en particular la cantidad de superficies), y pretende utilizar, en la medida de lo posible, la clase de superficies más sencillas de las que puede disponerse: las superficies planas.

Resulta interesante realizar las comparaciones de desempeño respecto al MSM de Mangasarian, particularmente, ya que lo que se plantea en este trabajo es que la principal limitación en ese método es *impedir la completa libertad* a las superficies generadas, restringiéndolas al paralelismo en cada iteración. Por esto, lo que diferencia a la instancia propuesta, de este método, simplemente *es el dar libertad a los hiper-planos generados*. Vale la pena agregar que tanto el MSM de Mangasarian como el propuesto, pueden considerarse algoritmos basado en árboles de decisión binarios, además de redes neurales. Por esto resulta poco interesante escoger otro algoritmo de árboles binarios para realizar comparaciones.

El foco de los experimentos, está entonces en distinguir la diferencia en la precisión de clasificación que puede deberse, estrictamente, al cambio en el modelo PL utilizado en cada iteración, que permite dar completa libertad a los hiper-planos generados.

4.2 Selección del Criterio de Comparación

El OGMS-LGP aparentemente genera superficies por partes más cercanas a lo que la intuición humana indicaría, en comparación con el MSM. Esto puede comprobarse en las Figuras 27, 28, 29 y 30, que muestran resultados típicos de ambos algoritmos. Sin embargo debe recordarse que la “mejor” hiper-superficie por partes ha de ser aquella que tenga mayor capacidad de generalización. Esto significa que la RNA generada debe ser la que tenga más efectividad a la hora de clasificar puntos “futuros” (en particular, puntos no utilizados durante el entrenamiento). Como lo importante de los puntos “futuros” es que son desconocidos (o asumidos como tales) en el momento del entrenamiento, se utilizó el método de Validación por 10 Grupos (*10-fold cross validation*). En este método, el conjunto de datos es dividido aleatoriamente en 10 conjuntos disjuntos de igual tamaño. El algoritmo en cuestión es entrenado con 9 de esas partes, a las que se denomina “conjunto de entrenamiento”. La 10ma parte, el “conjunto de validación”, es reservado para probar la precisión del modelo obtenido durante el entrenamiento. Esto es repetido 10 veces, utilizando en cada ocasión una de las partes como conjunto de validación. Se reportan tanto la precisión de clasificación promedio en los conjuntos de entrenamiento, como en los conjuntos de validación. La precisión de los conjuntos de validación es un estimado de la capacidad de generalización.

A medida que ambos algoritmos realizan sus iteraciones generando hiper-planos de separación, la precisión en el conjunto de entrenamiento por lo general tenderá a crecer, inclusive pudiendo llegar a ser de un 100%. Mientras tanto, la precisión en el conjunto de validación va a aumentar hasta un punto, en donde posiblemente la complejidad de la red esté originando un sobre-ajuste a los datos de entrenamiento, perdiendo la capacidad de clasificar correctamente a algunos de los puntos del conjunto de validación. En otras palabras, en ese momento la red empezará a perder capacidad de generalización. Por esto, los valores que se van a reportar serán los correspondientes a la iteración en la que se alcanza un máximo en la precisión promedio de los conjuntos de validación. Por ende, también se indicará el número de hiper-planos que por esta razón puede considerarse como el óptimo (que en principio ha de ser distinto para cada algoritmo y grupo de datos).

Con el propósito de tener una idea de la dispersión que puede tener el valor de la precisión, se reportan el valor máximo y mínimo alcanzados en las 10 corridas.

Puede ser de gran interés investigar con mayor profundidad la sensibilidad de la precisión respecto al valor que se escoja para el margen ϕ. En las formulaciones del capítulo anterior se tomó siempre como la unidad, y se le encuentra en el vector e presente en casi todas las restricciones, que no es más que un vector cuyos elementos son todos iguales a uno, con una dimensión escogida convenientemente de acuerdo al tamaño de las matrices A y B. También es interesante medir con más detalle la sensibilidad de la precisión respecto al valor de “permisividad” que se utilizó en la implementación, escogido como igual a ϕ como primera aproximación. Sobre estos aspectos se hablará en más detalle en la siguiente sección.

4.3 Diseño de los Experimentos

Se desea, en general, comprobar una mayor capacidad de generalización. En particular, también se desea comparar los dos algoritmos en cuanto a:

- Versatilidad ante datos “mal comportados”
- Semejanza con la intuición humana

Se han creado dos ejemplos sintéticos con datos "mal comportados". Los datos se crearon procurando las siguientes características:

- Que fuesen linealmente inseparables, motivando la ejecución de más de una iteración para ambos algoritmos.

- Que carecen de la conveniente y usual forma en que se presentan los datos cuando se toman en cuenta suficientes características para cada individuo. Esta forma a la que se hace referencia es la típica configuración en donde los cascos convexos de los dos conjuntos se interceptan en una zona difícil de describir, y a cada lado de ésta se encuentran, opuestas, las zonas mayoritariamente conformadas por puntos de uno de los dos conjuntos. El evitar esta configuración típica, enfrentando a los algoritmos a datos “mal comportados”, se hace resaltar las diferencias en la versatilidad de ambos algoritmos.

- Que pueden representarse cómodamente en dos dimensiones. Esto permite observar las características “geométricas” del comportamiento de ambos algoritmos. Es de interés particular evaluar la semejanza que puedan tener las superficies generadas con las que intuitivamente puede pensar un ser humano.

Las figuras 25 y 26 muestran la disposición *real* de los datos generados. Fueron denominados “Avión” y “4-Cuñas” para hacer referencia a ellos de forma breve y adicionalmente descriptiva de la configuración bi-dimensional de los elementos a clasificar.

Todos los datos utilizados para los experimentos fueron codificados para estar contenidos en un cuadrado centrado en cero cuyas coordenadas máxima y mínima son +1 y -1 respectivamente en ambas dimensiones. Por esta razón, los valores del “margen” que se utilizaron en la implementación de ambos algoritmos fueron ϕ=0,1 y ϕ=0,01, en lugar del propuesto inicialmente por Mangasarian y Bennett (ϕ=1). Los resultados numéricos muestran que en general los hiper-planos son generados por los modelos de Programación Lineal con mejor “calidad”, mientras menor es el valor del margen. En particular se sospecha que es importante que el "margen" sea por lo menos de un orden de magnitud más pequeño que el orden de magnitud del rango en que se encuentran los datos. En este sentido podrían realizarse estudios más profundos.

En la implementación se encontró que los valores de "margen" ϕ=0,001 o menores tienden a provocar “estancamiento” numérico en ambos algoritmos. Esto significa que recurrentemente el algoritmo falla en clasificar correctamente punto alguno, aún luego de utilizar los procedimientos de anti-degeneración implantados.

El fenómeno del estancamiento ocurrió ocasionalmente para ϕ=0,01 y ϕ=0,1, en ambos algoritmos. Esto se subsanó al introducir un ligero cambio al procedimiento de definición de los conjuntos A^{i+1} y B^{i+1}, introduciendo un factor de “permisividad”, del cuál se habló en la sección anterior. En el Algoritmo 6 se aprecia la modificación:

Este cambio en el procedimiento demostró inequívocamente mejor capacidad de generalización y comportamiento numérico razonable (el algoritmo no se estanca).

Sin embargo, más allá de las razones empíricas por las cuales resulta conveniente este cambio al procedimiento, existe un argumento válido para justificar el mismo.

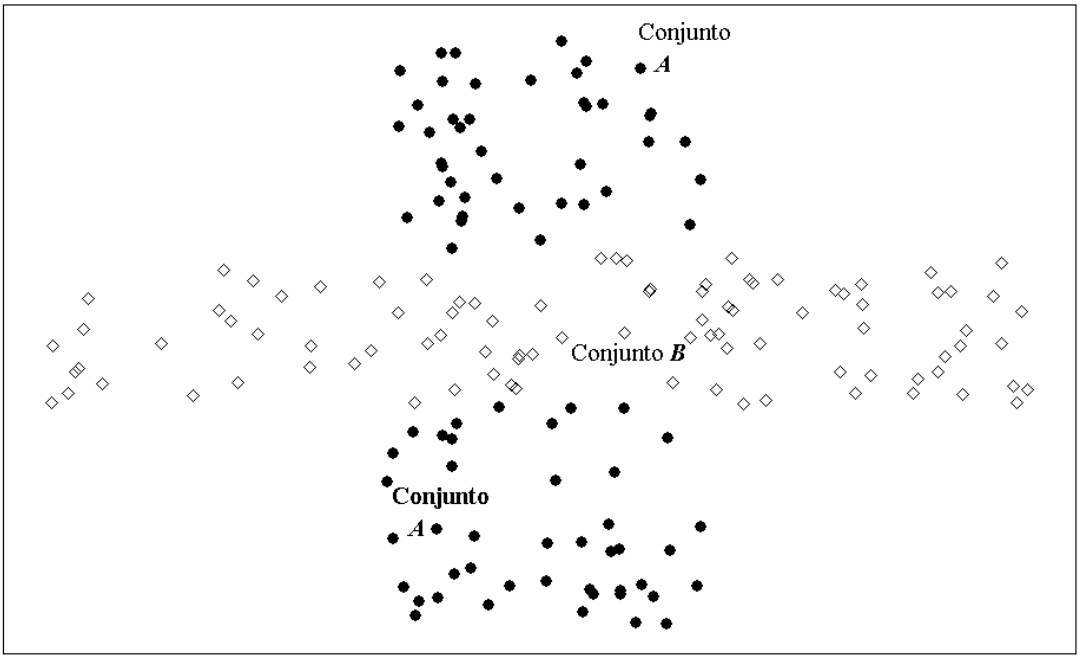

Figura 25: Conjunto de datos “Avión”

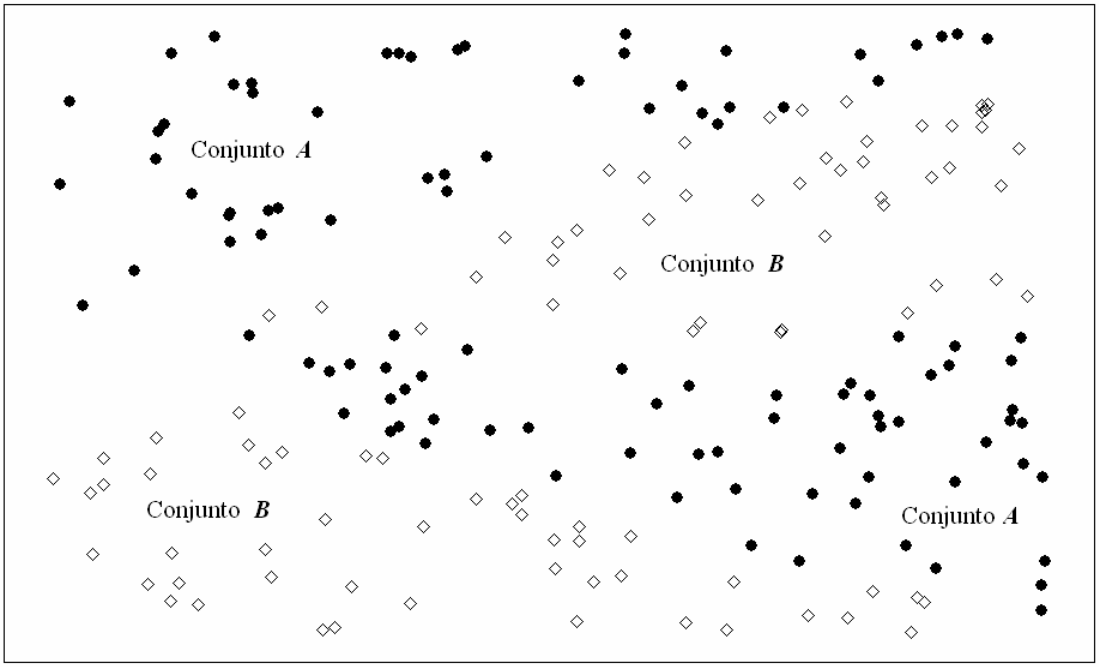

Figura 26: Conjunto de datos “4-Cuñas”

Definir los descendientes de $\mathcal{A}^i$ y $\mathcal{B}^i$:

$$A^{i+1}:=\{ A^i_m \mid A^i_m w^i_B \leq \theta^i_B + \phi \}$$

$$B^{i+1}:=\{ B^i_m \mid B^i_m w^i_A \geq \theta^i_A - \phi \}$$

Algoritmo 6: Modificación al procedimiento de definición de descendientes de $\mathcal{A}$ y $\mathcal{B}$

El ampliar el "margen de aceptación" de elementos para considerarlos como "ya clasificados", se está atendiendo al hecho de que en realidad es muy posible que estos hiper-planos estén sobre puntos que bien podrían considerarse clasificados. En especial, si el comportamiento de los datos no es muy típico, puede ocurrir que los únicos puntos que pueden "eliminarse" para pasar a la próxima iteración estén tan cerca del hiper-plano, que numéricamente estén sobre el mismo, ocasionando un "estancamiento". Ha de tenerse cuidado, sin embargo, en no ser demasiado permisivo con los puntos que se encuentren cerca del hiper-plano, pues el algoritmo podría estar descartando demasiados puntos sólo por esta razón y no porque el hiper-plano esté bien colocado. El exceso de permisividad implica un riesgo de sobre-estimación de la incertidumbre asociada a los datos que se pretende clasificar.

Se consideró que el tamaño de esa "permisividad" podría estar relacionado con el tamaño del "margen" ϕ. Por ello fue este el valor que se decidió utilizar. Después de todo el "margen" ϕ, por hipótesis, es un valor pequeño proporcional a la banda de "buena clasificación" de los hiper-planos calculados mediante optimización lineal. Está asociado a la "distancia mínima" que han de tener los puntos de un conjunto respecto al otro conjunto.

Los resultados numéricos revelaron una clase de soluciones que no habían sido reportadas en la literatura. La solución factible w=0, θ =-ϕ, y=0 (y=2ϕ) puede ser la solución óptima en el MSN de Mangasarian y en el OGMS-LGP. En ambos

algoritmos este tipo de resultados llamados por el autor como triviales (por lo no informativos), fueron evitados satisfactoriamente, en la mayoría de los casos, agregando al modelo la restricción $|w|_\infty = 1$.

4.4 Resultados Computacionales

Para analizar los datos generados sintéticamente en dos dimensiones, se implementó el algoritmo en Visual Basic para Aplicaciones, como una macro para Microsoft Excel, haciendo usando el complemento “SOLVER” para la resolución de los modelos de PL. Sin embargo, fue notorio que implementar el algoritmo de esta forma resultó ser muy complicado, debido al manejo de las celdas, rangos de celdas y referencias al Excel SOLVER. Los resultados obtenidos se muestran en la Tabla 1 y Tabla 2:

		Precisión (%)					
		OGMS-LGPP (MSN)			OGMS-LGP		
Conjunto de Datos		min	prom	Max	min	prom	Max
Avión	Entrenamiento	80,3	83,5	85,8	100	100	100
	Validación	68,4	82,2	95	94,7	99,5	100
4 Cuñas	Entrenamiento	59,3	67,7	71,8	79,1	90,1	92,7
	Validación	47,4	65,9	85	65	89	100

Tabla 1: Comparación de la precisión de clasificación del MSM con el OGMS-LGP

		Número “Óptimo” de Neuronas	
Conjunto de Datos		OGMS-LGPP (MSN)	OGMS-LGP
Avión	Entrenamiento	4	3
	Validación	4	3
4 Cuñas	Entrenamiento	14	4
	Validación	10	4

Tabla 2: Comparación del número de neuronas óptimo del MSM y el OGMS-LGP

Con el término “Precisión” se hace referencia a la proporción de elementos a clasificar que fueron correctamente clasificados por el algoritmo en cuestión. El número “óptimo” de neuronas fue determinado mediante la observación del comportamiento promedio de la precisión. El número de neuronas generadas en la iteración que presentase una mejor precisión promedio, fue el considerado como óptimo, en cada caso.

En las Figuras 27, 28, 29 y 30 se muestran geométricamente los resultados típicos *reales* de las superficies generadas por la primera iteración de ambos algoritmos, al aplicarse sobre los conjuntos de datos seleccionados,.

Puede apreciarse en las mismas que en todos los casos el OGMS-LGP genera superficies más cercanas a la intuición humana que las generadas por el MSM.

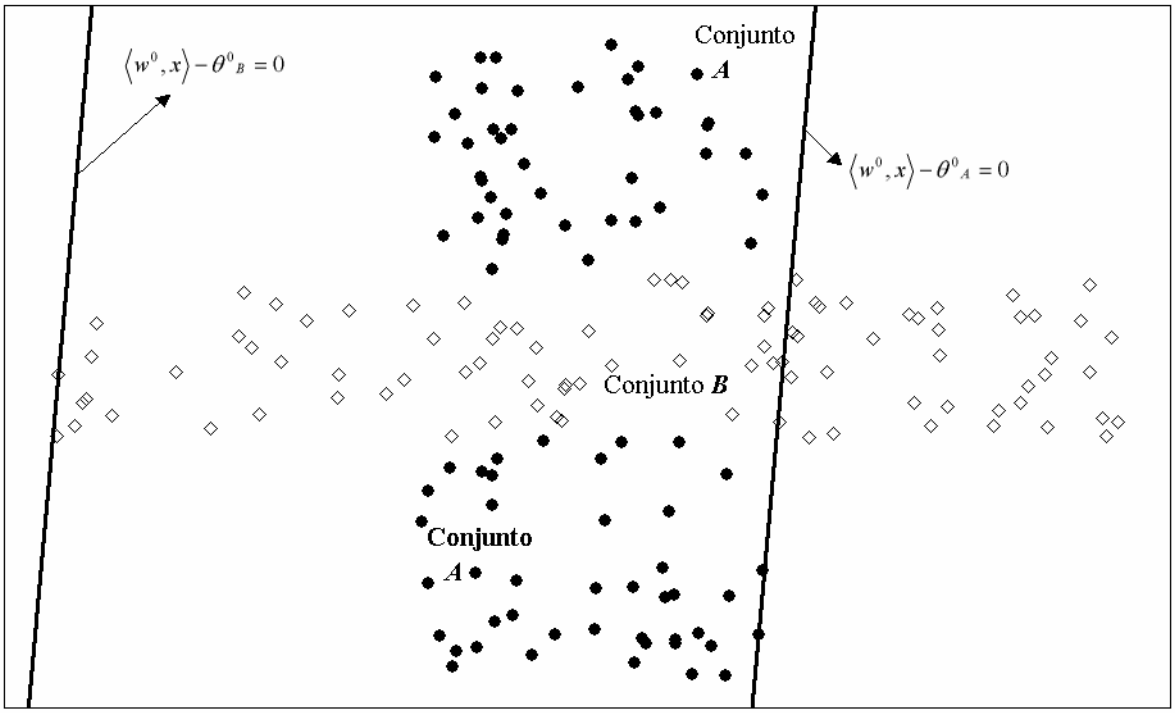

Figura 27: Típica 1ra iteración del MSM en el conjunto de datos “Avión”

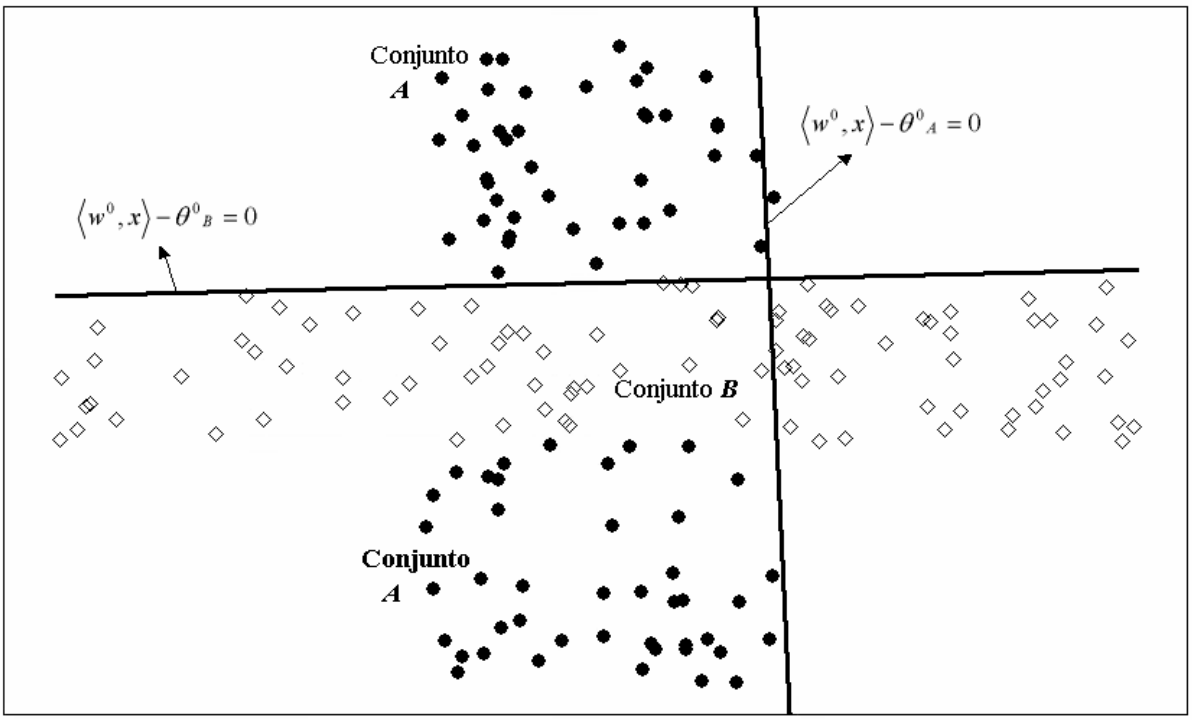

Figura 28: Típica 1ra iteración del OGMS-LGP en el conjunto de datos “Avión”

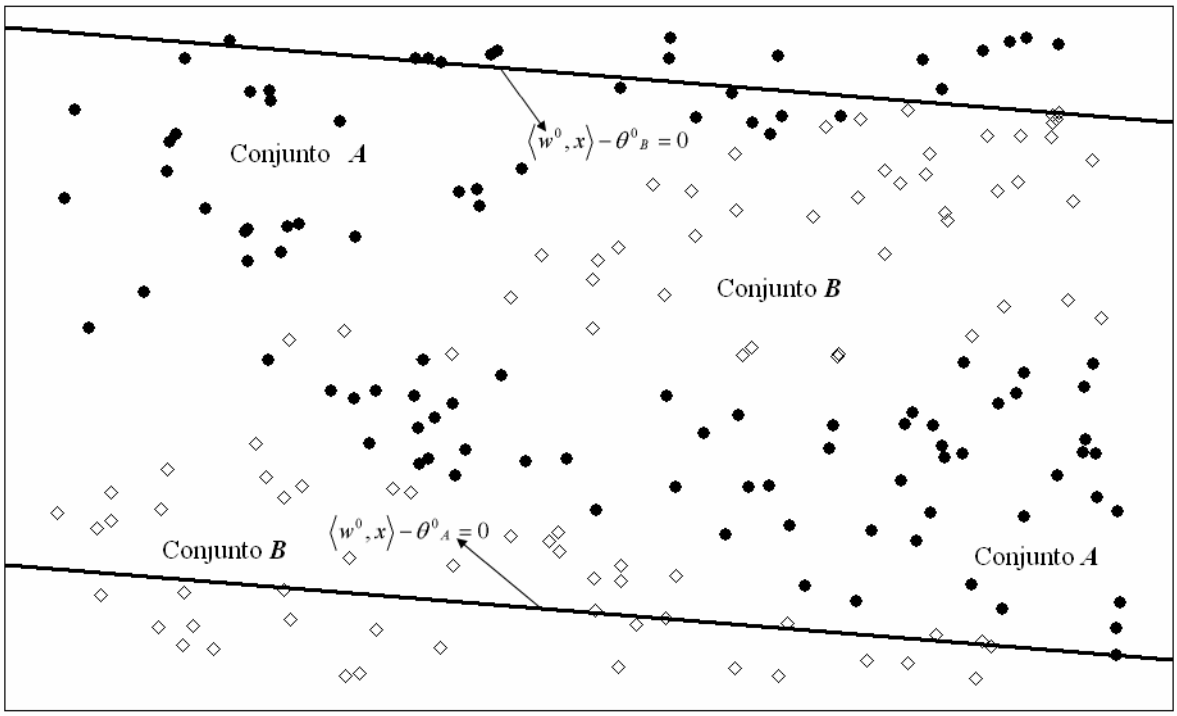

Figura 29: Típica 1ra iteración del MSM en el conjunto de datos “4-Cuñas”

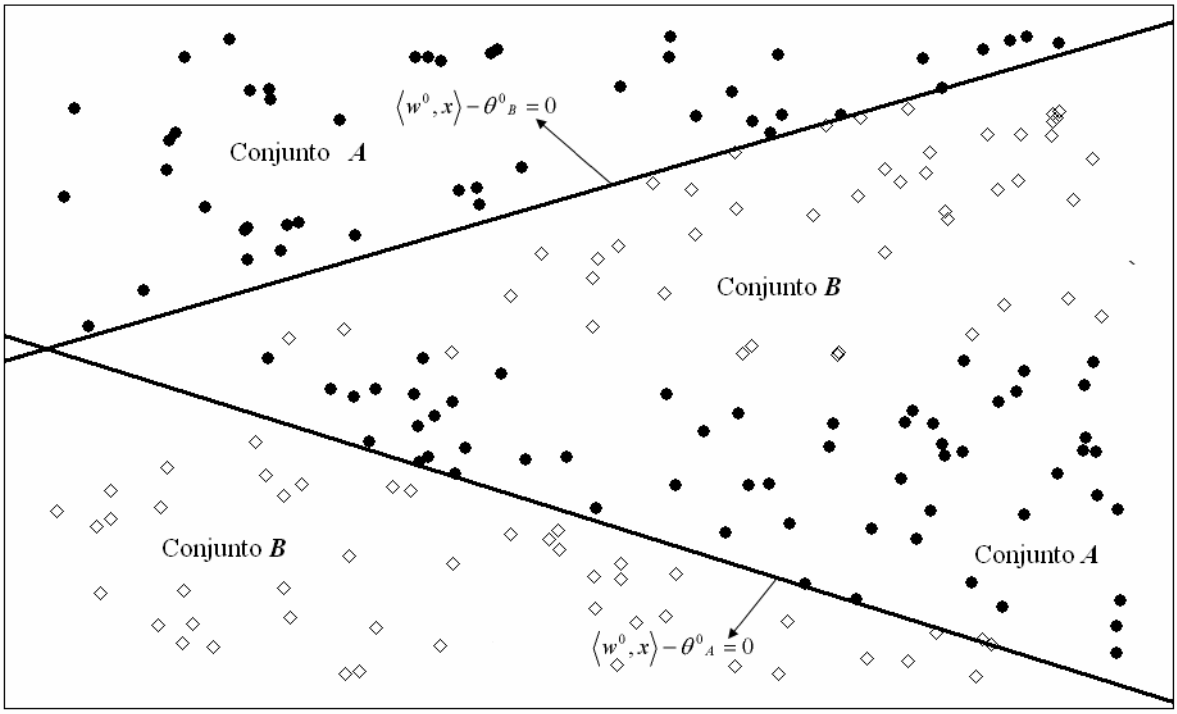

Figura 30: Típica 1ra iteración del OGMS-LGP en el conjunto de datos “4-Cuñas”

4.5 Conclusiones y Recomendaciones

Se ha estudiado el problema de entrenar y diseñar constructivamente un perceptrón multi-capa, evadiendo el uso del gradiente de la función empírica del riesgo de mala clasificación. En particular, partiendo del método propuesto por Mangasarian, se encontró una oportunidad de elaborar, no sólo un algoritmo con mejores propiedades, sino que también, en el proceso de análisis, se logró definir una familia de algoritmos. Esta fue creada procurando:

- alcanzar la mayor generalidad posible, y
- abrir campo para profundizar en diferentes instancias de esta familia.

Con la finalidad de verificar que las estrategias propuestas efectivamente consiguen una mejor capacidad de generalización, fue generada una instancia de la familia,

cuyo desempeño fue comparado con el del MSM de Mangasarian. Adrede se evitó hacer mayores cambios al algoritmo, salvo aquellos que implican el no-paralelismo de los hiper-planos, para que las causas de las diferencias encontradas en ambos niveles de desempeño no se confundiesen entre si. Por este motivo, otros criterios de optimalidad, y gestión de errores de clasificación fueron dejados para investigaciones posteriores.

Al realizarse pruebas en problemas prácticos, los resultados computacionales mostraron que la instancia propuesta mantuvo los beneficios de su algoritmo antecesor, evadiendo además la necesidad de procedimientos especiales para evitar la degeneración de los pares de hiper-planos.

Tal y como teóricamente podía sospecharse el desempeño del algoritmo fue, en la mayoría de los casos, significativamente superior. Nunca, ni en el peor de los casos, su desempeño fue peor que el de su algoritmo antecesor.

4.5.1 Aportes Adicionales

Como aportes adicionales al que se pretendió inicialmente, se tienen los siguientes, yendo de lo más general a lo más específico dentro del área de estudio:

a) Aunque algoritmos como el MSM han mostrado en la práctica sus grandes bondades, para el diseño y entrenamiento de RNAs, no ha habido mucho reconocimiento hacia este tipo de técnicas, en comparación con las técnicas clásicas asociadas a gradientes. Se plantea a la comunidad científica que trabaja en el área la siguiente incógnita: ¿cuáles son las razones que pueden encontrarse para justificar el uso de métodos asociados a gradientes, para realizar tareas que pueden prescindir de los problemas asociados a esos

métodos? Se espera que esta pregunta origine una interesante y productiva discusión.

b) Antes de la realización de este trabajo, no resultaban abundantes los algoritmos que cómo el de Mangasarian generaran de forma constructiva superficies lineales por partes para realizar tareas de clasificación binaria. Por esto, el nombre de Método Multi-Superficies (MSM por sus iniciales en inglés) bastaba para referirse al algoritmo de Mangasarian. Sin embargo, la definición del método general de Optimización Generativa de Multi-Superficies deja abiertas posibilidades respecto a cambios en el modelo de programación matemática que se utilice, los criterios de optimización, la estrategia de agregación de hiper-planos, entre otras, sin que por ellos esos métodos dejen de poder considerarse "Multi-Superficies". Esto permitió al autor sugerir una nomenclatura que permita nombrar a los algoritmos de manera más descriptiva. De hecho, bajo esta nomenclatura, el MSM de Mangasarian se re-bautiza como OGMS-LGPP (por las iniciales de Optimización Generativa Multi-Superficies Lineales Generadas a Pares Paralelos).

c) Se han identificado varias características que no resultan deseables en los métodos Multi-Superficie, tales como la pérdida de generalidad que implica el hecho de que los hiper-planos se trasladen hasta un punto en donde están "sobre" uno de los puntos a clasificar. También, asociado a ese tipo de estrategias, está el potencialmente excesivo gasto computacional asociado a la forma en que se calculan los pares de hiper-planos en el MSM, al ser construidos a partir de un presunto “mejor hiper-plano”, y desplazados en base a cálculos de valores máximos y mínimos de una cierta función. Sin embargo, este último argumento puede ser aplacado mediante el análisis del dual del modelo de optimización utilizado.

d) Se mostró que una de las instancias del método propuesto, idéntica a la implementada para su comparación con el MSM, pero exigiendo paralelismo a

los hiper-planos, no produce pares de hiper-planos degenerados, y adicionalmente lograría hiper-planos paralelos sin caer en las situaciones descritas como inconvenientes en el punto anterior.

e) Se reconoce la factibilidad de que ocurra una solución trivial para el modelo PL resuelto en cada iteración del MSM, que aún hasta la fecha no ha sido reportada, según es conocimiento del autor. La presente investigación sugiere el uso de la restricción adicional $|w|_\infty = 1$ para paliar este inconveniente.

4.5.2 Planes de Investigación

Como investigaciones futuras, se tiene planeado estudiar:

- El comportamiento de un algoritmo que genere, en cada iteración, sólo un hiper-plano, clasificando a elementos de un conjunto, posiblemente reduciendo la cantidad de restricciones que habrían de utilizarse para conseguir, en la siguiente iteración, un hiper-plano que clasifique elementos del otro conjunto.

- El diseño de modelos de PL que presenten mejor comportamiento en cuanto a evasión de hiper-planos que no clasifican punto alguno, así como soluciones triviales.

- La generalización de la Optimización Generativa de Multi-Superficies, para problemas de clasificación multi-categoría.

4.5.3 Recomendaciones

Entre las cosas que se concluyen como recomendaciones para futuro trabajo con algoritmos como el propuesto, están las siguientes:

- Estos algoritmos pueden utilizarse para diseñar y entrenar RNAs, pero no se recomiendan para trabajar en tiempo real. Se sugiere utilizar, para ajustar los parámetros de la RNA “en caliente”, algoritmos como la Retro-propagación con métrica variable, bajo la suposición de que los ajustes son pequeños.

- Se encontró que la implementación de este tipo de algoritmos en Visual Basic para Aplicaciones a manera de “macros”, resulta sustancialmente más complicada que en lenguajes como C++ y MATLab, debido a la dinámica de la referenciación del complemento SOLVER de Excel, y al manejo de celdas y sus rangos.

- Para escapar a soluciones triviales, la adición de la restricción $|w|_\infty = 1$, que obliga la resolución de $2n$ problemas (donde n es la dimensión del vector w), es la que presenta un mejor comportamiento. Su implementación para evitar resultados triviales en los modelos de optimización usados en el MSM y en el OGMS-LGP fue significativamente mejor que otras estrategias ensayadas.

Finalmente, se considera pertinente hacer un comentario sobre las implicaciones que conllevan los resultados obtenidos en este trabajo, desde una perspectiva más amplia.

El objeto de los estudios en RNAs es diferente, dependiendo de la inclinación de quien responda la pregunta. Para quienes vienen del área de la neuro-biología, el interés principal es entender más sobre la dinámica de las redes neurales biológicas, y cómo esta permite la realización de diversas funciones como la percepción, el

aprendizaje, la memoria asociativa, y el reconocimiento de patrones. Evidentemente hay muchas técnicas de entrenamiento y tipos de redes neurales que no se comportan como sus análogos biológicos. Sin embargo esto no preocupa a quienes enfocan el área de las RNAs desde el punto de vista ingenieril, ya que para estos, la dinámica de las redes neurales biológicas es esencialmente una inspiración para nuevas técnicas computacionales, con el objeto de resolver problemas prácticos.

Para quien realice la más breve investigación de los procesos neurales biológicos resultará claro que los algoritmos presentados acá no tienen un paralelo biológico, aunque pueden forzarse levemente las similitudes, con la intención de extraer algunas "moralejas" que puedan ser útiles para mejorar nuestras estrategias de resolución de problemas, como seres humanos.

Se podría pensar que el equivalente biológico a una RNA cuyo número de neuronas aumente a medida que el problema a resolver se muestra difícil de resolver, sería un cerebro que "genere" neuronas a medida que su ambiente se torna difícil de discernir, lo cual queda, evidentemente, en el campo de la ciencia ficción extrema. Las redes neurales naturales pueden ciertamente aumentar y reforzar la cantidad de conexiones que hay entre las neuronas, dependiendo de la complejidad de las tareas que a estas les compete resolver.

Sin embargo, existe una interpretación más interesante, y es la siguiente: en el cerebro cada neurona no sólo está dedicada a una tarea particular. Ella, en cambio, se asocia a diferentes grupos de neuronas para resolver problemas a cada momento. Por lo tanto, una misma neurona podría estar involucrada en varios procesos, o estar dedicada mayoritariamente a un proceso particularmente intensivo. Así, centrando el análisis en un proceso, y no en un grupo de neuronas, puede pensarse en que la mente, análogamente, podría ir asignando atención, recursos, o neuronas (dependiendo del nivel al que desee llevarse el símil), a esa tarea, a medida que la dificultad lo amerite. Sería un desperdicio dedicar toda la atención para un proceso relativamente simple. Sin embargo, si se desea resolver un nuevo problema, la

moraleja que podría desprenderse consiste en que no se realicen esfuerzos innecesariamente complejos, inicialmente, sino a medida que esto se justifique, pudiendo darse el caso de que toda la atención se centre en resolver esa tarea.

En otras palabras, intentar primero lo simple, y luego complicar las cosas, a medida que esto sea necesario.

El paralelo en el pensamiento humano, de la estrategia implícita en evitar el paralelismo de los hiper-planos, estaría en no asumir más estructura en los problemas, que la que ellos mismos sugieren. Esto puede complicar los procesos más de lo necesario, e inclusive lograr que la solución al problema eluda a nuestro entendimiento, gracias a una innecesaria parálisis paradigmática.

La moraleja es que *la no-flexibilidad en los procesos de aprendizaje (artificial) puede hacerlos ineficientes*. Queda abierta la discusión sobre la sensatez o no de la posible generalización de esa afirmación, generada en un contexto “artificial”, hacia el ámbito del aprendizaje “no-artificial”.

5. REFERENCIAS BIBLIOGRÁFICAS

[ASS00] Allwein E. L., Schapire R.E., Singer Y., “Reducing Multiclass to Binary: A Unifying Approach for Margin Classifiers”. Journal of Machine Learning Research, Vol.1 (2000), 113-141

[AG97] Amaldi E., Guenin B., “Two constructive methods for designing compact feedforward networks of threshold units”. International Journal of Neural Systems, Vol.8, Nos. 5 y 6 (1997), 629-645.

[BH89] Baum E.B., Haussler D., “What size net gives valid generalization?”. Neural Comput., Vol 1 (1989), 151-160.

[BBlue96] Bennett K., Blue J., “Optimal decision trees”. Technical Report No. 214, Departmentof Mathematical Sciences, Rensselaer Polytechnic Institute, Troy, New York (1996).

[BBred94] Bennet K., Bredensteiner E.J., “Global tree optimization: A non-greedy decision tree algorithm”. Comput. Sci. Stat., Vol.26 (1994), 156-160.

[BM92a] Bennett K.P., Mangasarian O.L., “Robust linear programming discrimination of two linearly inseparable sets¨. Optimization Methods and Software, Vol. 1 (1992), 23-34.

[BM92b] Bennett K.P., Mangasarian O.L., “Neural network training via linear programming”. P.M. Pardalos, editor, Advances in Optimization and Parallel Computing, Amsterdam, North Holland (1992), pp. 56-67.

[BM92c] Bennett K.P., Mangasarian O. L., “Bilinear separation of two sets in n-space”. Computer Sciences Department Technical Report 1109, University of Wisconsin, Madison, Wisconsin, 1992.

[BR89] Blum A., Rivest R.L., "Training a 3-node neural network is NP-Complete". En Touretzky D.S. Editor, Advances in Neural Information

Processing Systems I". 494-501, San Mateo, California, 1989. Morgan Kaufmann.

[BMS98] Bradley P.S., Mangasarian O.L., Street W.N., “Feature selection via mathematical programming”, INFORMS Journal on Computing, Vol.10, No.2 (1998), pp. 209-217.

[Bre95] Breiman L., “Reflections after refereeing papers for NIPS”. En Wolpert D.H., editor, “The Mathematics of Generalization”, Proceedings of the SFI/CNLS Workshop on Formal Approaches to Supervised Learning, Addison-Wesley Publishing Company, Santa Fe Institute Studies in the Sciences of Complexity, Vol. XX, publicado simultáneamente en E.E.U.U. y Canadá (1995), pp. 11-15.

[BFOS84] Breiman L., Friedman J.H., Olshen R.A., Stone C.J., “Classification and RegressionTrees”. Belmont, CA: Wadsworth (1984).

[BGP91] Burgess N., “A constructive algorithm that converges for real-valued input patterns”. Int. J. Neural Syst., Vol. 5 (1995), 59-66.

[CP95] Campbell C., Perez Vicente C., “The target switch algorithm: A constructive learning procedure for feedforward neural networks”, Neural Comput. Vol.7 (1995), 1245-1264.

[CFP97] Castellano G., Fanelli A.M., Pelillo M., “An iterative pruning algorithm for Feedforward Neural Networks”, IEEE Transactions on Neural Networks, Vol. 8, No.3 (1997), pp. 519-531.

[Char64] Charnes A., "Some fundamental theorems of perceptron theory and their geometry". En Lou J.T y Wilcox R.H, editores, Computer and Information Sciences, 67-74, Washington (1964), Spartan Books.

[CT92] Cohn D., Tesauro G., “How tight are the Vapnik-Chervonenkis bounds?” Neural Comput, Vol. 4 (1992), 249-269.

[Dix93] Dixon L.C.W., “Neural Networks and Unconstrained Optimization”. En Spedicato E., editor, “Algorithms for Continuous Optimization: The State of the Art”, NATO ASI Series C: Mathematical and Physical Sciences, Kluwer Academic Publishers, Vol. 434, Il Ciocco, Barga, Italy (1993), pp. 513-530.

[FL90] Fahlman S.E., Lebiere C., “The cascade-correlation learning arquitecture”. Advances in Neural Information Processing Systems 2, editado por Touretzky, San Mateo, C.A.: MorganKaufmann, (1990), 524-532.

[Gall86] Gallant S.I.,”Optimal linear discriminants”. Proceedings of the 8th International Conference on Pattern Recognition, Paris, Francia (1986), 849-852.

[Gar94] García Palomares U.M., "Aplicación de los métodos de proyección en el problema de factibilidad convexa: Un repaso". Investigación operativa, Vol. 4, No. 3 (1994), p. 229-245.

[Glo90] Glover F., "Improved linear programming models for discriminant analysis". Decision Sciences, No.21 (1990), 771-785.

[GGM04] González Castaño F.J., García Palomares U.M., Meyer R.R., "Projection support vector machine generators". Machine Learning 54-1 (2004), p. 33-44.

[Gri72] Grinold R.C., "Mathematical methods for pattern classification". Management Science, No. 19 (1972), 272-289.

[GGS01] Gutiérrez G., Galván I., Sanchis A., “Diseño de redes neuronales artificiales mediante computación evolutiva”. Reporte Técnico, Departamento de Informática, Universidad Carlos III de Madrid (2001).

[Hint89] Hinton G.E., “Connectionist learning procedures”. Artificial Intelligence, Vol.40, No.1 (1989), 143-150.

[HN95] Hinton G.E., Nowlan S., “Simplifying Neural Networks by Soft Weight Sharing”. En Wolpert D.H., editor, “The Mathematics of Generalization”, Proceedings of the SFI/CNLS Workshop on Formal Approaches to Supervised Learning, Addison-Wesley Publishing Company, Santa Fe Institute Studies in the Sciences of Complexity, Vol. XX, publicado simultáneamente en E.E.U.U. y Canadá (1995), pp. 11-15.

[HH91] Huang S.C., Huang Y.F., “Bounds on the number of hidden neurons in multiplayer perceptrons”. IEEE Transactions inNeural Networks, Vol.2 (1991), 47-55.

[IG04] Isasi Viñuela P., Galván León I.M., "Redes de Neuronas Artificiales. Un enfoque práctico". Pearson Educación, S.A., Madrid, 2004.

[JMM96] Jain A.K., Mao J., Mohiuddin K.M., “Artificial Neural Networks: A Tutorial”. Computer, Vol. 29, (1996).

[Judd90] Judd S.J., “Neural network design and the complexity of learning”. MIT Press, Cambridge, MA (1990).

[Lip87] Lippmann R.P., “An Introduction to Computing with Neural Nets”. IEEE ASSP Magazine, Abril (1987), 4-22.

[Man65] Mangasarian O.L., "Linear and nonlinear separation of patterns by linear programming". Operations Research, 13 (1965), 444-452.

[Man92] Mangasarian O.L., “Mathematical programming in neural networks”. Computer Sciences Department Technical Report 1129, University of Wisconsin, Madison, Wisconsin, 1992.

[Man68] Mangasarian O.L., “Multi-surface method of pattern separation”. IEEE Transactions on Information Theory, IT-14 (1968), 801-807.

[MSW90] Mangasarian O.L., Setiono R., Wolberg W.H., "Pattern recognition via linear programming: Theory and application to medical diagnosis". En Coleman T.F. y Li Y. editores, Proceedings of the Workshop on Large-Scale Numerical Optimization, Cornell University, Ithaca, Nueva York, 10-20 de Octubre de 1989, pp. 22-23, Philadelphia , Pennsylvania. SIAM (1990).

[MG93] Marchand M., Golea M., “On learning simple neural concepts: From halfspace intersections to neural decision lists”. Network: Computation in Neural Systems, Vol.4 (1993), 67-85.

[McCP43] McCulloch W., Pitts W., “A logical calculus of the ideas immanent in nervous activity”. Bulletin of Mathematical Biophysics, No.7 (1943), 115-133.

[Megi88] Megiddo N., "On complexity of polyhedral separability". Discrete and Computational Geometry, No. 3 (1988), 325-337.

[MN89] Mézard M., Nadal J.-P., “Learning in feedforward layered networks: The Tiling algorithm”. J. Phys. A, Vol.22 (1989), 2191-2204.

[MTH89] Miller G., Todd P., Hegde S., “Designing neural networks using genetic algorithms”. Third International Conference on Genetic Algorithms, editado por Shaffer J. (1989), 379-384.

[MP69] Minsky M., Papert S., “Perceptrons: an introduction to computational geometry”. MIT Press, Cambridge, Massachusetts, 1969.

[MS54] Motzkin T.S., Schoenberg I.J., “The relaxation method for linear inequalities”. Canadian Journal of Mathematics, No.6 (1954), 393-404.

[MMRTS01] Müller K.R., Mika S., Rätsch G., Tsuda K., Schölpf B., “An introduction to kernel-based learning algorithms”. IEEE Transactions on Neural Networks, Vol. 12, No.2 (2001) 181-202.

[MA92] Murphy P.M., Aha D.W., “UCI repository of machine learning databases”. Department of Information and Computer Science, University of Califormia, Irvine, California (1992).

[RWW94] Rohwer R., Wynne-Jones M., Wysotzki F., “Neural Networks”. En Michie D., Spiegelhalter D.J. y Taylor C.C. editores, “Machine learning: Neural and statistical classification”, Ellis Horwood Series in Artificial Intelligence, Gran Bretaña (1994), pp. 84-114.

[Ros62] Rosenblatt F., “Principles of Neurodynamics”. Spartan Books, New York, New York, 1962.

[Ros57] Rosenblatt F., “The perceptron- a perceiving and recognizing automaton”. Technical Report 85-460-1, Cornell Aeronautical Laboratory, Itahca, New York, January 1957.

[RM90] Roy A., Mukhopadhyay S., "Pattern classification using linear programming", ORSA Journal of Computing, No. 3 (1990), 66-80.

[SD88] Sietsma J., Dow R.J.F., “Neural net prunning: Why and how”. Proceedings of the IEEE International Conference in Neural Networks, San Diego, CA, Vol.1 (1988), 325-333.

[SN90] Sirat J.A., Nadal J.-P., “Neural Trees: A new tool for classification”. Network: Computation in Neural Systems, Vol.1 (1990), 423-438.

[Smi68] Smith F.W., "Pattern classifier design by linear programming". IEEE Transactions on Computers, C-17 (1968), 367-372.

[Vap98] Vapnik V.N., “The Nature of Statistical Learning Theory”, Springer, 2nd Edition, New York, 193 pp, 1998.

[WM90] Wolberg W.H., Mangasarian O.L., "Multisurface method of patterns separation for medical diagnosis to breast cytology". Proceedings of the National Academy of Science, E.E.U.U., No.87 (1990), 9193-9196.

[Wol95] Wolpert D.H., “The Status of Supervised Learning Science Circa”. En Wolpert D.H. editor, “The Mathematics of Generalization”, Proceedings of the SFI/CNLS Workshop on Formal Approaches to Supervised Learning, Addison-Wesley Publishing Company, Santa Fe Institute Studies in the Sciences of Complexity, Vol. XX, publicado simultáneamente en E.E.U.U. y Canadá (1995), pp. 1-10.

[YL97] Yao X., Lin Y., “A new evolutionary system for evolving artificial neural networks”. IEEE Transactions on Neural Networks, Vol. 8, No. 3 (1997), 694-713.

[Yu92] Yu X.-H., “Can backpropagation error surface not have local minima”. IEEE Transactions in Neural Networks, Vol.3 (1992), 1019-1021.