INSTITUTO POLITECNICO NACIONAL

CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN

APLICACIÓN DE LOS MODELOS ASOCIATIVOS ALFA-BETA A LA BIOINFORMÁTICA

T E S I S

QUE PARA OBTENER EL GRADO DE
MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA:

ISRAEL ROMÁN GODÍNEZ

DIRECTOR DE TESIS:

DR. CORNELIO YÁÑEZ MÁRQUEZ

MÉXICO, D.F. JUNIO DE 2007

Dedico el presente trabajo

A DIOS, elemento fundamental de todo el que hacer diario mío y de mi familia

A mis Padres, JOEL Y YOLANDA, quienes con su ejemplo guían el camino de mis hermanos y mío.

A mis hermanos, JOEL Y RODRIGO, con los que comparto, felizmente, una vida de sueños, acciones e ilusiones.

A mis ABUELOS, tanto paternos como maternos, quienes son el más alto eslabón de la cadena que es mi familia.

A mis Tíos, en ambas familias,, cuyos consejos y apoyo son importantes en cada una de mis metas.

A mis AMIGOS, la familia que se encuentra cuando se esta lejos del hogar.

Agradecimientos

El poder terminar mi maestría así como esta tesis se lo debo a muchas personas e instituciones, en especial quiero agradecer a las siguientes:

A mi familia y amigos, que siempre se acuerdan de mí y me brindan su cariño.

A mi asesor, el *Dr. Cornelio Yáñez Márquez*, por todo lo que me ha enseñado y por su apoyo constante en el trabajo realizado para esta tesis.

A mis sinodales, los doctores *Sergio Suárez, Oleksiy Pogrebnyak, Oscar Camacho, Maria Elena Acevedo* y el maestro *Marco Moreno*, por la sabiduría, consejo y paciencia que me ofrecieron en la conclusión de esta tesis.

Al *Instituto Politécnico Nacional* y *CONACYT* por el apoyo económico que me otorgaron durante estos dos años.

Al *Centro de Investigación en Computación* por abrirme las puertas a una nueva experiencia.

Índice detallado de la tesis

Índice de figuras

Índice de tablas

Índice de ecuaciones

Resumen

En este trabajo de tesis se desarrolla un algoritmo para clasificación de patrones que funciona de manera notable con bases de datos de Bioinformática. El modelo de memorias asociativas sobre el que está basado dicho algoritmo es el Alfa-Beta.

Para poder alcanzar un buen porcentaje de clasificación, fue necesario desarrollar un nuevo algoritmo de Memorias Heteroasociativas que permitiera recuperar el conjunto fundamental completo, propiedad que por si sola representaría una aportación significativa. Otra de las aportaciones hechas es la multimemoria heteroasociativa Alfa-Beta, la cual es la base del algoritmo de clasificación presentado en esta tesis. Además, se desarrolló el correspondiente sustento matemático, el cual nos permite conocer el comportamiento del nuevo modelo propuesto y así poder predecir fallas y aciertos de nuestro sistema, aun antes de crearlo.

Para verificar la eficacia tanto del nuevo algoritmo de Memorias Heteroasociativas como del clasificador de patrones, se desarrolló un software que nos permite realizar evaluaciones del algoritmo al aplicar técnicas de validación con patrones pertenecientes a bases de datos de Bioinformática.

Con el desarrollo de este trabajo de tesis se muestra la importancia que hasta ahora han adquirido las Memorias Asociativas en las áreas de aprendizaje y recuperación de patrones, debido a su capacidad para representar el conocimiento de manera compacta.

Abstract

In this thesis we develop a pattern classifier algorithm that works notably with bioinformatics data bases. The associative memories model in which the classifier is based on is the Alfa-Beta associative memories.

In order to achieve a good classification average it was necessary to develop a new heteroassociative memory algorithm that let us recall completely the fundamental set. Property that would be a significative contribution by itself. In addition, a heteroasociative multimemory Alfa-Beta is created, as a fundamental base for the proposed classifier. Moreover, the corresponding mathematical foundation let us known the behavior of the new proposed model and that way to predict fails on a system before create it.

The efficiency of the new heteroassociative memories algorithm and the good response of the new pattern classifier are verified by software. This software let us asses the algorithm applying validation techniques to instances of the bioinformatics data base.

This thesis show the importance that the associative memories have achieved this days in the learning and recalling patterns due to their ability to represent the knowledge in compact way.

CAPÍTULO 1. Introducción

n esta tesis se hace una modificación al modelo original de las memorias heteroasociativas Alfa-Beta, y se crea un algoritmo de clasificación de patrones, el cual se aplica en problemas de bioinformática, específicamente en área de *promotores* y *zonas de empalme*, con un rendimiento notable y competitivo respecto de algoritmos actuales.

1.1 Antecedentes

La biología molecular ha tenido avances significativos en las últimas décadas; avances tan importantes que, para su estudio, ha sido necesario echar mano no sólo de herramientas propias, sino de otras disciplinas, como es el caso de la ciencia de la computación.

Uno de los métodos computacionales más utilizados en el ámbito de la Bioinformática (como se le conoce a la ciencia multidisciplinaria que fusiona la biología con la computación) es el de las *Redes Neuronales Artificiales.* Algunas de las principales tareas que se han intentado resolver mediante el uso de esta técnicas, son problemas de *clasificación y funciones de aproximación*.

Dadas las características que poseen las redes neuronales, algunos de los principales problemas que se han intentado resolver son la alineación de secuencias, análisis proteínicos, localización de genes, *identificación de promotores y determinación de zonas de empalme* siendo estas dos últimas las tareas que se atacan en este trabajo.

El tema de las memorias asociativas ha estado vigente desde hace varios lustros, dentro de algunas áreas de la investigación debido a su sencillez en la aplicación y su gran potencialidad. La característica más importante y al mismo tiempo el propósito fundamental de una memoria asociativa, es recuperar correctamente patrones completos a partir de patrones de entrada, los cuales pueden estar alterados con ruido aditivo, sustractivo o combinado [1, 2, 19].

Una memoria asociativa está constituida básicamente por dos fases principales: la *fase de aprendizaje* que es el proceso mediante el cual se crea la memoria asociativa a través de la formación de asociaciones de patrones, y la *fase de recuperación* en la cual se le presentan a la memoria, patrones que pueden ser del conjunto fundamental o no, esperando obtener el correspondiente patrón asociado. Si en cada asociación sucede que el patrón de entrada es igual al de salida, la memoria es autoasociativa; en caso contrario, la memoria es heteroasociativa: esto significa que una memoria autoasociativa puede considerarse como un caso particular de una memoria heteroasociativa [2].

En los últimos 45 años se han desarrollado modelos de memorias asociativas paralelamente a las Redes Neuronales, desde la concepción del primer modelo de neurona artificial [3], hasta los modelos de redes neuronales basados en conceptos modernos como la morfología matemática [4]. En el lapso entre estos dos modelos se fueron desarrollando importantes trabajos de los pioneros en redes neuronales tipo *perceptron* [5, 6, 7] y el modelo Hopfield, presentado al mundo en 1982, con la característica principal de funcionar como red neuronal y como memoria asociativa [8]. Con este trabajo de investigación, Hopfield propició el resurgimiento de las redes neuronales después del período posterior a la publicación del libro *Perceptrons* [9], en donde los autores demostraron que el *perceptron* tenía severas limitaciones.

El primer modelo de memoria asociativa conocido apareció 21 años antes que el modelo de Hopfield, y se debe al científico alemán Karl Steinbuch, quien en 1961 desarrolla una memoria heteroasociativa que funciona como un clasificador de patrones binarios: la *Lernmatrix* [10]. Ocho años después, los investigadores escoceses Willshaw, Buneman y Longuet-Higgins [11] presentan el *Correlograph*, dispositivo óptico elemental capaz de funcionar como una memoria asociativa. Dos modelos clásicos de memorias asociativas fueron presentados en 1972 por Anderson [12] y Kohonen [13] de manera independiente, y debido a su importancia y a la similitud de los conceptos involucrados, ambos modelos reciben el nombre genérico de *Linear Associator*.

Al modelo Hopfield, sin afán de demeritar su importancia, se le pueden detectar dos grandes desventajas: la primera es el hecho de que sólo es autoasociativo, por lo que no es capaz de asociar patrones diferentes; y en segundo lugar, es notable el hecho de que la capacidad de recuperación de patrones es muy pequeña, sólo de $0.15n$ siendo n la dimensión de los patrones almacenados [8].

A partir de estas fechas comienza una carrera por crear nuevos modelos o mejorar los ya existentes. Uno de los primeros intentos fue el realizado en 1988 por Bart Kosko [14] quien creó un modelo de memoria heteroasociativa a partir de la memoria Hopfield: la memoria asociativa bidireccional BAM (Bidirectional Associative Memory) la cual, al igual que la memoria Hopfield, se basa en un algoritmo iterativo. A pesar de que el intento de Kosko fue exitoso al obtener una memoria heteroasociativa, la segunda desventaja de la memoria Hopfield no fue subsanada con la BAM: la memoria asociativa bidireccional de Kosko exhibe una muy baja capacidad de aprendizaje y recuperación de patrones, al igual que los modelos subsecuentes de memorias asociativas bidireccionales [15-18].

Posteriormente a la creación de estos modelos, surge, al final del siglo pasado en la década de los noventa, lo que actualmente conocemos como *memorias asociativas morfológicas*. Dichas memorias se crearon a partir de que el científico Gerhard X. Ritter y su equipo de investigación aplicaron los conceptos de la morfología matemática y de las redes neuronales morfológicas [4] para diseñar un nuevo tipo de memorias asociativas que lograron superar las capacidades de aprendizaje y recuperación de patrones de todos los modelos conocidos hasta ese momento [19].

Las memorias asociativas morfológicas sirvieron de inspiración para la creación de las *memorias asociativas Alfa-Beta* desarrolladas en el año 2002 por investigadores del Centro de Investigación en Computación del IPN [20]. Este modelo, entre cuyos resultados más relevantes está el haber servido de fundamento teórico para la creación de un nuevo tipo de memorias asociativas bidireccionales con máxima capacidad de aprendizaje y almacenamiento, las BAM Alfa-Beta [21-23] constituye, en gran medida, la base teórica del presente trabajo de tesis.

Por otro lado, así como las memorias asociativas son un intento del hombre por simular el comportamiento de la memoria humana, de la misma forma se ha intentado simular la capacidad humana de reconocer objetos [24, 25]; a esta área importante en la computación se le conoce como *Reconocimiento de Patrones.*

Existen diversos enfoques o metodologías susceptibles de implementarse en las computadoras, y cada uno de estos enfoques está formado por una serie de algoritmos denominados clasificadores, que resuelven el problema de forma parcial, algunos con mayor éxito que otros. El sustento de cada uno puede variar avanzadas teorías matemáticas hasta los basados fuertemente en técnicas heurísticas [24, 25, 26].

Las memorias asociativas son reconocidas por su capacidad de reconocer patrones, por lo que son consideradas como una de las metodologías de esta rama. La principal razón para ello es que tiene como objetivo recuperar correctamente patrones completos a partir de patrones de entrada los cuales pueden o no estar alterados por ruido [8, 13, 19, 27, 28, 39]. Es a raíz de esta aseveración que nace el nuevo enfoque de clasificación de patrones, *el enfoque asociativo* con la creación de un nuevo clasificador: Clasificador Hibrido de patrones basado en la Lernmatrix de Steinbuch y el Linear Associator de Anderson-Kohonen conocido también como CHAT (Clasificador Hibrido Asociativo con Translación) [40]. Posteriormente surge el segundo clasificador dentro de este enfoque, basado principalmente en la Lernmatrix de Steinbuch, y tiene un desempeño muy similar al del CHAT pero con un algoritmo más simple [41].

1.2 Justificación

Dados los avances científicos que se han realizado en el ámbito de la biología molecular, y a la gran cantidad de información que se ha generado a partir de ellos, ha surgido la

necesidad de procesar dicha información de una manera más rápida y con la misma efectividad, o mejor, que la de un experto. De aquí surge una nueva ciencia denominada *Bioinformática*, campo multidisciplinario que fusiona, entre otras, dos importantes ramas de la ciencia: *biología molecular* y *computación* [43], con el principal objetivo de aprovechar las ventajas evidentes que ofrece el campo de las ciencias de la computación, en virtud de su enorme velocidad de desarrollo, eficiencia y eficacia de los algoritmos [45].

El desarrollo de bases de datos, la alineación de secuencias proteínicas, la secuenciación de cadenas de DNA y la predicción en la estructura de proteínas, clasificación de estructuras proteínicas, *identificación de promotores*, *localización de zonas de empalme* (splice-junction) determinación de relaciones filogenéticas, se pueden mencionar entre los primeros y principales problemas que entran en el ámbito de esta ciencia [44-46].

Algunos de los problemas antes mencionados tienen su base en la búsqueda y localización de patrones dentro de ciertas secuencias, que permita clasificarlas; un ejemplo de estas tareas son la *identificación de promotores* y *localización de zonas de empalme (splice-junction)*, que como ya se menciono, será parte del tema de estudio de este trabajo de tesis. Es por ello que el contar con algoritmos confiables y de alto rendimiento es una necesidad imperiosa para esta importante rama de la ciencia [47].

Las memorias asociativas, y en especial las Alfa-Beta son, por lo tanto, una buena herramienta computacional debido a la simplicidad de los algoritmos, su fuerte sustento matemático, y alta efectividad para el reconocimiento de patrones.

1.3 Objetivo

Desarrollar un algoritmo que permita modificar el modelo original de las memorias heteroasociativas Alfa-Beta, de modo que el reconocedor de patrones creado a partir del nuevo algoritmo sea capaz de llevar a cabo tareas de clasificación de patrones en problemas de Bioinformática.

1.4 Objetivos específicos

Para alcanzar el objetivo de la tesis, es preciso cumplir los siguientes objetivos específicos:

- Modificación del modelo original de las memorias heteroasociativas Alfa-Beta, para que recuperen el conjunto fundamental completo.
- Creación del algoritmo de multimemorias heteroasociativas.
- Desarrollo de un algoritmo que permite la creación de un clasificador de patrones de rendimiento notable, al ser aplicado a problemas de Bioinformática.

1.5 Organización del documento

En este capítulo se presentaron: los antecedentes, justificación, objetivos general y específicos relacionados con esta tesis. El resto del documento está organizado de la siguiente manera:

En el capítulo 2 se presentan dos secciones, la primera trata sobre los conceptos básicos y principales problemas atacados en Bioinformática; y la segunda nos introduce tanto en los conceptos básicos de las memorias asociativas como en los modelos representativos anteriores a las memorias asociativas Alfa-Beta.

El capítulo 3 se enfoca tanto en el modelo de memorias asociativas Alfa-Beta como en promotores y zonas de empalme (splice-junction), estos últimos, problemas relevantes en Bioinformática.

El capítulo 4 presenta los nuevos algoritmos, creados a partir de las memorias asociativas Alfa-Beta, que serán la base y fundamentan la descripción del nuevo clasificador de patrones que se aplicará en Bioinformática, dicho algoritmo también es explicado en este capítulo. Cabe mencionar que en este capítulo se encuentra la parte esencial de la tesis.

Los resultados experimentales obtenidos a partir de una base de datos conocida en Bioinformática es incluida en el capítulo 5, así como las discusiones correspondientes en relación con el nuevo algoritmo. El capítulo 6 presenta las conclusiones y el trabajo a futuro y, finalmente, se anexan dos apéndices: Apéndice A (manual de usuario del software) y Apéndice B Simbología; y las referencias bibliográficas.

CAPÍTULO 2. Estado del arte

En este capítulo se presenta el estado del arte, tanto de la bioinformática, como de las memorias asociativas. En la sección 2.1 se describen los conceptos básicos de Bioinformática y se presentan los principales problemas atacados en esta ciencia, además de algunos ejemplos. En la sección 2.2 se describen los conceptos básicos de memorias asociativas y se muestran, además, los modelos más representativos anteriores a las memorias asociativas Alfa-Beta, dado que este modelo se presentará en el capítulo 3.

1.1 Bioinformática

En las últimas décadas, el desarrollo de la tecnología genómica ha ocasionado la generación de una gran cantidad de información, que a su vez nos conduce a un requerimiento, cada vez mayor, de recursos computacionales para su almacenamiento, organización, recuperación y principalmente visualización y análisis [50].

La Bioinformática puede ser definida en palabras simples como: “El uso de métodos computacionales para hacer descubrimientos biológicos”, concepto que visto desde otra perspectiva es el campo interdisciplinario que engloba la biología, ciencias de la computación, matemáticas y estadística para analizar secuencias biológicas, contenido genético, y predecir funciones y estructuras de macromoléculas [43].

Acorde con [47] la meta mas reciente de la bioinformática es *permitir el descubrimiento de nuevos elementos biológicos que apoyen la creación de una perspectiva global de la cual se puedan derivar principios biológicos uniformes.*

1.1.1 Conceptos básicos

El *Ácido Desoxirribonucleico (ADN)* y las *proteínas* son macromoléculas biológicas que están constituidas por una larga cadena de componentes químicos. El ADN está formado por un conjunto de elementos denominados *nucleótidos* o *bases*; la secuencia del ADN juega un papel fundamental en los diferentes procesos bioquímicos de los organismos vivos, de los cuales los principales: 1) contienen los moldes para la síntesis de proteínas y 2) funcionan como medio de transmisión de la información entre padres e hijos [48].

Los nucleótidos consisten de una base nitrogenada, una molécula de azúcar (desoxirribosa) y un fosfato. Las cuatro bases nitrogenadas son denotadas por la primera letra de cada base: A (adenina), C (citosina), G (guanina) y T (timina). Los nucleótidos de una cadena de ADN están unidos a través de las moléculas de fósforo y azúcar formando una cadena larga que constituye una cadena de ADN. Cada una de estas cadenas tiene la propiedad de estar unida a otra cadena conocida como cadena complementaria, esto a raíz de cierta característica de los nucleótidos de establecer parejas especificas, A-T y G-C. La unión de estos dos tipos de cadenas complementarias da como resultado una cadena que tiene forma de doble hélice, la cual fue propuesta por primera vez por Watson y Crick en 1953. Este tipo de configuración asegura que cada cadena contiene la información necesaria, y la maquinaria bioquímica para que la información pueda ser copiada de generación en generación [49].

Las cadenas en la doble hélice son antiparalelas; además, la dirección a lo largo de cada cadena está determinada por la posición del átomo de carbono en el anillo de desoxirribosa que, unido a los fosfatos, crea las cadenas. Dependiendo de la numeración de los carbonos, si la cadena comienza con el quinto carbono se dice que la secuencia va de 5’ a 3’; y para el caso contrario, si comienza con el tercer carbono la dirección va en el sentido 3’ a 5’ [47-51].

Al ADN se encuentra alojado en las unidades básicas que constituyen a los seres vivos, las células. Los seres vivos pueden ser clasificados, en organismos *procariontes* o *eucariontes* dependiendo el tipo de células que los constituyen. Las células procariontes son las más sencillas y pequeñas, y tienen la característica de que carece de núcleo, entre ellas podemos encontrar las bacterias. La mayoría de las células procariontes contienen su material genético como elementos circulares independientes menores a 5 Mbp de longitud y este se encuentra esparcido en toda la célula. Por otro lado, existen organismos que contienen un núcleo más organizado, las células eucariontes; estas células, a diferencia de las primeras, contienen un núcleo definido [48, 50]. En la mayoría de las células eucariontes, el ADN se encuentra alojado en el núcleo, separado en paquetes denominados *cromosomas*. Cada cromosoma contiene una molécula simple de la cadena de ADN con forma de doble hélice. Pequeñas cantidades de ADN aparecen fuera del núcleo y se alojan en la mitocondria y el cloroplasto [50].

Dentro de las cadenas de ADN existen ciertas cadenas de nucleótidos conocidas como *genes*. Los genes son secuencias de tripletas de nucleótidos, también llamadas *exones*, las cuales tienen una correspondencia particular con una proteína. Los exones que forman a los genes están localizados en las cadenas de ADN pero en forma desordenada; es decir, no es posible tomar una porción de la cadena y aseverar que ésta codificará, tal cual, a una proteína. Entre cada exon existen secuencias que no codifican una proteína, y a estas secuencias se les conoce como *intrones*; a las secuencias que tampoco codifican proteínas y que se encuentran entre cada gen, se le denomina *regiones intergenéticas* [49, 50, 51].

Las proteínas están compuestas de 20 diferentes aminoácidos, los cuales están denotados por 20 diferentes letras. Cada uno de los 20 aminoácidos es codificado por uno o más codones [48]. Las propiedades químicas que diferencian a los 20 aminoácidos causa que éstos se agrupen en, y conformen, cierta estructura 3-D que define las funciones especificas de una célula [51].

1.1.2 Principales problemas en bioinformática

Los diferentes problemas analizados por la bioinformática pueden ser clasificados dentro de dos categorías: *tareas genómicas* y *tareas proteómicas*. El término *tareas genómicas* es usado para denotar el estudio de varios genomas viéndolos como entidades que presentan contenidos similares. Por otro lado, el termino *tareas proteómicas*, hace referencia al estudio de todas las proteínas que surgen a partir de un genoma [84].

Tareas Genómicas

Análisis de secuencias genéticas

El *alineamiento de secuencias* está basado en los principios de *similitud* y *homología* [52]. Similitud es una medida cuantitativa entre dos genes que son idénticos en términos de cantidades observables y homología se refiere a como dos secuencias, que pueden ser muy similares o no, comparte un ancestro en común.

Alineamiento de secuencia

Un alineamiento es un arreglo de dos o más secuencias que exhiben dónde las secuencias son similares y dónde no lo son. El alineamiento *óptimo* es aquel que exhibe el mayor número de correspondencias y el menor número de diferencias. Básicamente existen dos tipos de métodos de alineamiento, *alineamiento global* y *alineamiento local*; el primero maximiza el número de coincidencias entre las secuencias a lo largo de la cadena completa, y el segundo selecciona el número más alto de coincidencias en cierta porción de la secuencia [53, 54].

Búsqueda de patrones

Se buscan patrones nucleicos en una secuencia de ácido nucleico, en un conjunto de secuencias o en una base de datos. Esto es importante para descubrir relaciones evolutivas o patrones entre diferentes formas de vida [55].

Localización de genes e identificación de promotores

Debido a que la secuencia de ADN es demasiado larga, es necesaria la identificación automática de genes de una larga secuencia de ADN. Las células tienen por sí mismas un mecanismo que reconoce el comienzo de un gen, y a este identificador se le conoce como *promotor*. Los promotores son regiones anteriores a cada gen que indican que enseguida se encontrará uno. Los promotores son los elementos que regulan el inicio de la transcripción [56].

Tareas Proteómicas

Las proteínas son polipéptidos formados dentro de las células como cadenas de aminoácidos [57]. Existen diferentes tareas relacionadas con el análisis de proteínas; a continuación se mencionan las más importantes.

Alineación múltiple de secuencia

Técnicas de alineación múltiple de secuencia de aminoácidos son utilizadas con los siguientes objetivos: a) encontrar el mayor número de secuencias coincidentes en diferentes secuencias alineadas, b) ayudar a la predicción de estructuras secundarias y terciarias de nuevas secuencias, y c) servir como paso preliminar en el análisis de la evolución molecular usando métodos filogenéticos [43].

Búsqueda de motifs

El objetivo de la búsqueda de *motifs* es encontrar patrones específicos en secuencias de proteínas. Las proteínas son generadas de un conjunto básico de bloques a los que conocemos como dominios. A través de los años, el proceso de evolución ha ido cambiando estos dominios, dando origen a una gran diversidad de secuencias proteínicas, y dado que parten de un conjunto básico, es posible encontrar que proteínas distintas compartan cierta relación local o global. Es por esta situación que la búsqueda de motifs es una técnica que busca en secuencias ciertos dominios que tengan un significado biológico que categorizará a la proteína en cierta familia [43,51].

Genómica Estructural

La genómica estructural se encarga de la predicción de estructuras tridimensionales de proteínas a partir de una secuencia primaria de aminoácidos [58], siendo ésta una de las tareas más complicadas en Bioinformática. El método de similitud puede ser usado para la predicción de estructuras secundarias y terciarias basado en su homología.

Expresión de genes y Microarreglos

Expresión genética

La expresión de genes es el proceso por el cual la información que codifica a un gen es convertida en una estructura funcional de las células. La expresión de genes incluye la transcripción de una secuencia genética en su correspondiente mRNA y su traslación a proteína, además de aquellas secuencias que se transcriben en RNA pero que no es convertido en proteínas. Por lo tanto, la *expresión genética* estudia los niveles de expresión de los genes en las células bajo diferentes condiciones [59].

Microarreglos

La tecnología de los microarreglos permite medir los niveles de expresión de miles de genes al mismo tiempo. Para este tipo de análisis lo ideal es que cada una de las moléculas contenidas en los microarreglos identifique un gen o un exon del genoma. Esta técnica nos ayuda para comparar la expresión de genes en su estado normal contra células con cierta enfermedad, por ejemplo células cancerígenas [60].

Análisis de las redes reguladoras de genes.

Una pregunta muy importante en la biología es ¿cómo se inicia o se detiene la expresión de genes?; es decir, cómo son regulados los genes [43]. Debido a que casi todas las células contienen el mismo contenido genómico, es posible deducir que la diferencia entre los procesos celulares no radica en el contenido genómico sino en la expresión genómica. La regulación en los genes no ha sido entendida completamente, pero es evidente que un tipo de proteína llamado *factor de transcripción* juega un papel muy importante. Esta proteína se adhiere a partes específicas del ADN conocidas como *sitios obligatorios de factor de transcripción* que están localizados en zonas conocidas como *regiones de promotores*.

Promotores específicos son asociados con un gen en particular y generalmente no están tan lejanos de sus respectivos genes. Los factores de transcripción controlan la expresión de genes colocando los genes promotores en las regiones de promotores activando o reprimiendo la transcripción del mismo [59,64].

1.1.3 Métodos computacionales aplicados en Bioinformática

Uno de los métodos computacionales más utilizados en el ámbito de la Bioinformática son las *Redes Neuronales Artificiales;* dichos modelos tratan de emular la red neuronal biológica. Algunas de las tareas en las que son más utilizadas las redes neuronales artificiales son en problemas de *clasificación* y *funciones de aproximación*. Dado que las memorias asociativas son un modelo de red neuronal y que no se han realizado muchos trabajos en bioinformática aplicando memorias asociativas para la resolución de problemas, a continuación se muestran aportes hechos principalmente con redes neuronales.

Alineamiento de secuencias

Uno de los modelos más conocidos en redes neuronales, la *red neuronal de Hopfield*, fue aplicada para el alineamiento molecular. Para este caso las moléculas fueron representadas por cuatro tipos de propiedades químicas, y con estas propiedades se crea una correspondencia entre ellas. Este método es aplicado para el análisis de relaciones estructura-actividad tridimensional [61].

Otra de las redes neuronales creadas fue GenTHREADER. La arquitectura de esta red neuronal predice la similitud entre dos secuencias genéticas. Esta red neuronal es utilizada principalmente en la *predicción de estructuras proteínicas* obteniendo buenos resultados [62].

También, se ha utilizado el algoritmo *backpropagation* para entrenar redes neuronales que se aproximan a los resultados obtenidos por una de las principales herramientas de similitud utilizadas en Bioinformática *BLAST*, obteniendo una ventaja sobre BLAST en lo que respecta a la velocidad de procesamiento; sin embargo, en la comparación de resultados, la red neuronal queda por debajo de la herramienta BLAST [62-66].

Localización de genes e identificación de promotores

El trabajo realizado por Lukashin *et al.* [63] es una de las primeras investigaciones hechas para el reconocimiento de localización de promotores en secuencias de ADN utilizando redes neuronales artificiales. El proceso utiliza un subconjunto pequeño, tomado aleatoriamente, del total de las muestras de secuencias de promotores (del orden del 10%) para llevar a cabo el entrenamiento de la red. La capacidad de aprendizaje es puesta a prueba al presentarle el total del conjunto de secuencias, obteniendo una efectividad de entre el 94 al 99%.

En lo que respecta a la localización de genes, se han realizado análisis cuantitativos de similitud entre secuencias de genes tRNA, principalmente para la búsqueda de relaciones evolutivas. Secuencias nuevas han sido introducidas a esta red neuronal y se han reconocido como genes tRNA [64].

Otro ejemplo relacionado con la localización de promotores es una red neuronal multicapa feed-forward cuya arquitectura es entrenada para predecir si una secuencia de nucleótidos es una secuencia promotora bacterial [65].

Análisis de proteínas

Las redes neuronales se han convertido en la técnica más efectiva para la predicción de estructuras proteínicas tridimensionales; la predicción se lleva a cabo mediante la alineación de secuencias de proteínas desconocidas con un conjunto de estructuras proteínicas homólogas. Algunos de los trabajos realizados son los siguientes:
La investigaciones realizadas por Qian *et al*. [66] es una de las primeras en este sentido. Para el entrenamiento se utilizaron más de 100 estructuras proteínicas. Después del entrenamiento, cuando la red neuronal fue probada, se obtuvo un porcentaje de predicción del 64%.

Posteriormente, en lugar de usar una sola secuencia proteínica para probar a la red neuronal previamente entrenada, se crearon *perfiles* mediante la alineación múltiple de secuencias, presentando este perfil como entrada de la red neuronal y obteniendo resultados del 71.4% [67, 68].

También las redes neuronales *backpropagation* son utilizadas para la predicción de estructuras proteínicas secundarias. Wood *et al.* compararon el rendimiento de red neuronal con arquitectura cascada-correlación con una *backpropagation*, observando que la primera obtenía resultados muy similares a los de la segunda pero en un menor tiempo [69, 70].

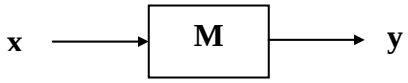
1.2 Memorias Asociativas

En esta sección se describen los conceptos básicos de memorias asociativas y se muestran, además, los modelos más representativos anteriores a las memorias asociativas Alfa-Beta, dado que este modelo se presentará en el capítulo 3

1.2.1 Conceptos básicos

Los conceptos presentados en esta sección se han tomado de las referencias que, a nuestro juicio, son las más representativas [1, 2, 20, 21,38].

Una *memoria asociativa* **M** puede visualizarse como un sistema de entrada y salida, donde los patrones de entrada y salida están representados por vectores columna denotados **x** y **y** respectivamente. A continuación se muestra un esquema de la idea anterior.



Cada uno de los patrones de entrada forma una asociación con el correspondiente patrón de salida. Tal asociación es similar a la una pareja ordenada; por ejemplo, los patrones **x** y **y** del esquema anterior forman la asociación (**x**,**y**).

No obstante que a lo largo de este capítulo se manejaran algunas de las notaciones originales establecidas en los distintos modelos, a continuación se presenta la nomenclatura que será manejada en lo sucesivo, para la descripción de los conceptos básicos sobre las memorias asociativas y el resto de esta tesis.

Los patrones de entrada y salida se denotarán con las letras negrillas, **x** y **y**, agregándoles números naturales como superíndices para efectos de discriminación simbólica. Por ejemplo, a un patrón de entrada $\mathbf{x}^1$ le corresponderá el patrón de salida $\mathbf{y}^1$, y ambos formarán la asociación $(\mathbf{x}^1,\mathbf{y}^1)$; del mismo modo, para un número entero positivo k específico, la asociación correspondiente será $(\mathbf{x}^k,\mathbf{y}^k)$.

La memoria asociativa **M** se representa mediante una matriz, la cual se genera a partir de un conjunto finito de asociaciones conocidas de antemano, a este nuevo conjunto se le conoce como **conjunto fundamental de aprendizaje**, o simplemente **conjunto fundamental**.

El conjunto fundamental se representa de la siguiente manera:

$$\{(\mathbf{x}^{\mu}, \mathbf{y}^{\mu}) \mid \mu = 1, 2, ..., p\}$$

donde p es un número entero positivo que representa la cardinalidad del conjunto fundamental.

A los patrones que conforman las asociaciones del conjunto fundamental se les llama **patrones fundamentales**. La naturaleza del conjunto fundamental proporciona un importante criterio para clasificar las memorias asociativas:

Una memoria es **Autoasociativa** si se cumple que $\mathbf{x}^{\mu} = \mathbf{y}^{\mu} \ \forall \mu \in \{1, 2, ..., p\}$, por lo que uno de los requisitos que se debe de cumplir es que $n = m$. De otra manera si $\exists \mu \in \{1, 2, ..., p\}$ para el que se cumple que $\mathbf{x}^{\mu} \neq \mathbf{y}^{\mu}$ se trata de una memoria **Heteroasociativa**. Nótese que puede haber memorias heteroasociativas con $n = m$.

Con el fin de especificar las componentes de los patrones, se requiere dejar en claro cual será la notación para dos conjuntos a los que llamaremos arbitrariamente A y B. Las componentes de los vectores columna que representan a los patrones, tanto de entrada como de salida, serán elementos del conjunto A, y las entradas de la matriz **M** serán elementos del conjunto B.

No hay requisitos previos ni limitaciones respecto de la elección de estos dos conjuntos, por lo que no necesariamente deben ser diferentes o poseer características especiales. Esto significa que el número de posibilidades para escoger A y B es infinito.

Como se aclaro previamente, cada uno de los modelos de memorias asociativas que se presentan en este capítulo tiene sus propias especificaciones para dichos conjuntos, de acuerdo a las necesidades del creador.

Por convención, cada vector columna que representa a un patrón de entrada tendrá n componentes cuyos valores pertenecen al conjunto A, y cada vector columna que representa a un patrón de salida tendrá m componentes cuyos valores pertenecen también al conjunto A. Es decir:

$$\mathbf{x}^{\mu} \in A^n \text{ y } \mathbf{y}^{\mu} \in A^m \ \forall \mu \in \{1, 2, ..., p\}$$

La j-ésima componente de un vector columna se indicará con la misma letra del vector, pero sin negrilla, colocando a j como subíndice ($j \in \{1, 2, ..., n\}$ o $j \in \{1, 2, ..., m\}$ según corresponda). La j-ésima componente del vector columna $\mathbf{x}^{\mu}$ se representa por: x_j^{μ}

En los problemas donde intervienen las memorias asociativas, se consideran dos fases importantes: La fase de aprendizaje, que es donde se genera la memoria asociativa a partir de las p asociaciones del conjunto fundamental, y la fase de recuperación que es donde la memoria asociativas opera sobre un patrón de entrada, a la manera del esquema que aparece al inicio de este capítulo.

1. **Fase de Aprendizaje** (Generación de la memoria asociativa). Encontrar los operadores adecuados y una manera de generar una matriz **M** que almacene las p asociaciones del conjunto fundamental $\{(\mathbf{x}^1,\mathbf{y}^1), (\mathbf{x}^2,\mathbf{y}^2), ..., (\mathbf{x}^p,\mathbf{y}^p)\}$, donde $\mathbf{x}^{\mu} \in A^n$ y

$\mathbf{y}^{\mu} \in A^m \ \forall \mu \in \{1, 2, ..., p\}$. Si $\exists \mu \in \{1, 2, ..., p\}$ tal que $\mathbf{x}^{\mu} \neq \mathbf{y}^{\mu}$, la memoria será heteroasociativa; si $m = n$ y $\mathbf{x}^{\mu} = \mathbf{y}^{\mu} \quad \forall \mu \in \{1, 2, ..., p\}$, la memoria será autoasociativa.

2. **Fase de Recuperación** (Operación de la memoria asociativa). Hallar los operadores adecuados y las condiciones suficientes para obtener el patrón fundamental de salida $\mathbf{y}^{\mu}$, cuando se opera la memoria **M** con el patrón fundamental de entrada $\mathbf{x}^{\mu}$; lo anterior para todos los elementos del conjunto fundamental y para ambos modos: autoasociativo y heteroasociativo.

Se dice que una memoria asociativa **M** exhibe **recuperación correcta** si al presentarle como entrada, en la fase de recuperación, un patrón $\mathbf{x}^{\omega}$ con $\omega \in \{1, 2, ..., p\}$, ésta responde con el correspondiente patrón fundamental de salida $\mathbf{y}^{\omega}$.

Este último punto es muy importante ya que el hecho de que la memoria asociativa tenga recuperación perfecta aumenta, ampliamente, la gama de posibilidades en donde es posible aplicar a las memorias asociativas.

1.2.2 Principales Modelos de Memorias Asociativas

En esta sección presentaremos una breve reseña de los modelo de memorias asociativas, con el objeto de establecer el marco de referencia sobre el que se basan la memorias asociativas alfa-beta.

Los modelos principales sobre los que se abordará son: *Lernmatrix*, *Correlograph*, *Linear Associator*, *Memoria Hopfield*, *Memorias Asociativas Morfológicas*, *Memorias Asociativas Alfa-Beta* y *Memorias Asociativas Mediana*, siendo los cuatro primeros modelos basados en el anillo de los números racionales con las operaciones de multiplicación y adición, el antepenúltimo modelo esta basado sobre las operaciones morfológicas y el paradigma de suma de productos y el penúltimo utiliza máximos y mínimos de dos operaciones nuevas expresadas en [20] conocidas como operación Alfa y operación Beta y el último basado en nuevas operaciones sustentadas en [42].

Lernmatrix de Steinbuch

Karl Steinbuch fue uno de los primeros investigadores en desarrollar un método para codificar información en arreglos cuadriculados conocidos como *crossbar* [20]. La importancia de la *Lernmatrix* [10, 29] se evidencia en una afirmación que hace Kohonen [13] en su artículo de 1972, donde apunta que las matrices de correlación, base fundamental de su innovador trabajo, vinieron a sustituir a la *Lernmatrix* de Steinbuch.

La *Lernmatrix* es una memoria heteroasociativa que puede funcionar como un clasificador de patrones binarios si se escogen adecuadamente los patrones de salida; es un sistema de entrada y salida que al operar acepta como entrada un patrón binario $\mathbf{x}^{\mu} \in A^n, \quad A = \{0,1\}$

y produce como salida la clase $\mathbf{y}^{\mu} \in A^{p}$ que le corresponde (de entre *p* clases diferentes), codificada ésta con un método que en la literatura se le ha llamado *one-hot* [30].

La codificación *one-hot* funciona así: para representar la clase $k \in \{1, 2, ..., p\}$, se asignan a las componentes del vector de salida $\mathbf{y}^{\mu}$ los siguientes valores: $y_k^{\mu} = 1$, y $y_j^{\mu} = 0$ para $j = 1, 2, ..., k\text{-}1, k\text{+}1, ..., p$.

Algoritmo de la Lernmatrix

Fase de Aprendizaje

Se genera el esquema (*crossbar*) al incorporar la pareja de patrones de entrenamiento $\left(\mathbf{x}^{\mu}, \mathbf{y}^{\mu}\right) \in A^{n} \times A^{p}$. Cada uno de los componentes m_{ij} de **M**, la *Lernmatrix* de Steinbuch, tiene valor cero al inicio, y se actualiza de acuerdo con la regla $m_{ij} + \Delta m_{ij}$, donde:

$$\Delta m_{ij} = \begin{cases} +\varepsilon & \text{si } x_j^{\mu} = 1 = y_i^{\mu} \\ -\varepsilon & \text{si } x_j^{\mu} = 0 \text{ y } y_i^{\mu} = 1 \\ 0 & \text{en otro caso} \end{cases} \qquad \textbf{1.1}$$

donde ε una constante positiva escogida previamente: es usual que ε es igual a 1.

Fase de Recuperación

La *i*-ésima coordenada y_i^{ω} del vector de clase $\mathbf{y}^{\omega} \in A^{p}$ se obtiene como lo indica la siguiente expresión, donde $\vee$ es el operador *máximo*:

$$y_i^{\omega} = \begin{cases} 1 & \text{si } \sum_{j=1}^{n} m_{ij} . x_j^{\omega} = \bigvee_{h=1}^{p} \left[\sum_{j=1}^{n} m_{hj} . x_j^{\omega} \right] \\ 0 & \text{en otro caso} \end{cases} \qquad \textbf{1.2}$$

Correlograph de Willshaw, Buneman y Longuet-Higgins

El *correlograph* es un dispositivo óptico elemental capaz de funcionar como una memoria asociativa [11, 31]. En palabras de los autores “el sistema es tan simple, que podría ser construido en cualquier laboratorio escolar de física elemental”.

Algoritmo del Correlograph

Fase de Aprendizaje

La *red asociativa* se genera al incorporar la pareja de patrones de entrenamiento $(\mathbf{x}^{\mu}, \mathbf{y}^{\mu}) \in A^{n} \text{ x } A^{m}$. Cada uno de los componentes m_{ij} de la *red asociativa* **M** tiene valor cero al inicio, y se actualiza de acuerdo con la regla:

$$m_{ij} = \begin{cases} 1 & \text{si } y_i^{\mu} = 1 = x_j^{\mu} \\ \textit{valor anterior} & \text{en otro caso} \end{cases} \qquad \textbf{1.3}$$

Fase de Recuperación

Se le presenta a la *red asociativa* **M** un vector de entrada $\mathbf{x}^{\omega} \in A^n$. Se realiza el producto de la matriz **M** por el vector $\mathbf{x}^{\omega}$ y se ejecuta una operación de umbralizado, de acuerdo con la siguiente expresión:

$$y_i^{\omega} = \begin{cases} 1 & \text{si } \sum_{j=1}^{n} m_{ij}.x_j^{\omega} \geq u \\ 0 & \text{en otro caso} \end{cases} \qquad \textbf{1.4}$$

donde *u* es el valor de umbral. Una estimación aproximada del valor de umbral *u* se puede lograr con la ayuda de un número indicador mencionado en el artículo [11] de Willshaw *et al.* de 1969: $log_2 n$.

Linear Associator de Anderson-Kohonen

El *Linear Associator* tiene su origen en los trabajos pioneros de 1972 publicados por Anderson y Kohonen [12, 13, 36].

Para presentar el *Linear Associator* consideremos de nuevo el conjunto fundamental:

$$\{(\mathbf{x}^{\mu}, \mathbf{y}^{\mu}) \mid \mu = 1, 2, ..., p\} \text{ con } A = \{0, 1\}, \quad \mathbf{x}^{\mu} \in A^n \quad \text{y} \quad \mathbf{y}^{\mu} \in A^m$$

Algoritmo del Linear Associator

Fase de Aprendizaje

1) Para cada una de las *p* asociaciones $(\mathbf{x}^{\mu}, \mathbf{y}^{\mu})$ se encuentra la matriz $\mathbf{y}^{\mu} \cdot \left(\mathbf{x}^{\mu}\right)^t$ de dimensiones *m* x *n*

2) Se suman la *p* matrices para obtener la memoria

$$\mathbf{M} = \sum_{\mu=1}^{p} \mathbf{y}^{\mu} \cdot \left(\mathbf{x}^{\mu}\right)^t = \left[m_{ij}\right]_{m \times n} \qquad \textbf{1.5}$$

de manera que la *ij*-ésima componente de la memoria M se expresa así:

$$m_{ij} = \sum_{\mu=1}^{p} y_i^{\mu} x_j^{\mu} \qquad \textbf{1.6}$$

Fase de Recuperación

Esta fase consiste en presentarle a la memoria un patrón de entrada $\mathbf{x}^{\omega}$, donde $\omega \in \{1, 2, ..., p\}$ y realizar la operación

$$\mathbf{M} \cdot \mathbf{x}^{\omega} = \left[\sum_{\mu=1}^{p} \mathbf{y}^{\mu} \cdot \left(\mathbf{x}^{\mu} \right)^{t} \right] \cdot \mathbf{x}^{\omega} \qquad \textbf{1.7}$$

La memoria asociativa Hopfield

El artículo de John J. Hopfield de 1982, publicado por la prestigiosa y respetada *National Academy of Sciences* (en sus *Proceedings*), impactó positivamente y trajo a la palestra internacional su famosa memoria asociativa [8].

En el modelo que originalmente propuso Hopfield, cada neurona x_i tiene dos posibles estados, a la manera de las neuronas de McCulloch-Pitts: $x_i = 0$ y $x_i = 1$; sin embargo, Hopfield observa que, para un nivel dado de exactitud en la recuperación de patrones, la capacidad de almacenamiento de información de la memoria se puede incrementar por un factor de 2, si se escogen como posibles estados de las neuronas los valores $x_i = -1$ y $x_i = 1$ en lugar de los valores originales $x_i = 0$ y $x_i = 1$.

Al utilizar el conjunto {-1, 1} y el valor de umbral cero, la fase de aprendizaje para la memoria Hopfield será similar, en cierta forma, a la fase de aprendizaje del *Linear Associator*. La intensidad de la fuerza de conexión de la neurona x_i a la neurona x_j se representa por el valor de m_{ij}, y se considera que hay simetría, es decir, $m_{ij} = m_{ji}$. Si x_i no está conectada con x_j entonces $m_{ij} = 0$; en particular, no hay conexiones recurrentes de una neurona a sí misma, lo cual significa que $m_{ij} = 0$. El estado instantáneo del sistema está completamente especificado por el vector columna de dimensión n cuyas coordenadas son los valores de las n neuronas.
La memoria Hopfield es autoasociativa, simétrica, con ceros en la diagonal principal. En virtud de que la memoria es autoasociativa, el conjunto fundamental para la memoria Hopfield es $\{(\mathbf{x}^{\mu}, \mathbf{x}^{\mu}) \mid \mu = 1, 2, ..., p\}$ con $\mathbf{x}^{\mu} \in A^n$ y $A = \{-1, 1\}$

Algoritmo Hopfield

Fase de Aprendizaje

La fase de aprendizaje para la memoria Hopfield es similar a la fase de aprendizaje del *Linear Associator*, con una ligera diferencia relacionada con la diagonal principal en ceros, como se muestra en la siguiente regla para obtener la ij-ésima componente de la memoria Hopfield **M**:

$$m_{ij} = \begin{cases} \sum_{\mu=1}^{p} x_i^{\mu} x_j^{\mu} & \text{si } i \neq j \\ 0 \text{ si } i = j \end{cases} \qquad \textbf{1.8}$$

Fase de Recuperación

Si se le presenta un patrón de entrada $\tilde{\mathbf{x}}$ a la memoria Hopfield, ésta cambiará su estado con el tiempo, de modo que cada neurona x_i ajuste su valor de acuerdo con el resultado que arroje la comparación de la cantidad $\sum_{j=1}^{n} m_{ij} x_j$ con un valor de umbral, el cual normalmente se coloca en cero.

Se representa el estado de la memoria Hopfield en el tiempo t por $\mathbf{x}(t)$; entonces $x_i(t)$ representa el valor de la neurona x_i en el tiempo t y $x_i(t+1)$ el valor de x_i en el tiempo siguiente $(t+1)$.

Dado un vector columna de entrada $\tilde{\mathbf{x}}$, la fase de recuperación consta de tres pasos:

1) Para $t = 0$, se hace $\mathbf{x}(t) = \tilde{\mathbf{x}}$; es decir, $x_i(0) = \tilde{x}_i$, $\forall i \in \{1,2,3,...,n\}$
2) $\forall i \in \{1,2,3,...,n\}$ se calcula $x_i(t+1)$ de acuerdo con la condición siguiente:

$$x_i(t+1) = \begin{cases} 1 \text{ si } \sum_{j=1}^{n} m_{ij} x_j(t) > 0 \\ x_i(t) \text{ si } \sum_{j=1}^{n} m_{ij} x_j(t) = 0 \\ -1 \text{ si } \sum_{j=1}^{n} m_{ij} x_j(t) < 0 \end{cases} \qquad \textbf{1.9}$$

3) Se compara $x_i(t+1)$ con $x_i(t)$ $\forall i \in \{1, 2, 3,...,n\}$. Si $\mathbf{x}(t+1) = \mathbf{x}(t)$ el proceso termina y el vector recuperado es $\mathbf{x}(0) = \tilde{\mathbf{x}}$. De otro modo, el proceso continúa de la siguiente manera: los pasos 2 y 3 se iteran tantas veces como sea necesario hasta llegar a un valor $t = \tau$ para el cual $x_i(\tau+1) = x_i(\tau)$ $\forall i \in \{1, 2, 3,...,n\}$; el proceso termina y el patrón recuperado es $\mathbf{x}(\tau)$.

En el artículo original de 1982, Hopfield había estimado empíricamente que su memoria tenía una capacidad de recuperar $0.15n$ patrones, y en el trabajo de Abu-Mostafa & St. Jacques [32] se estableció formalmente que una cota superior para el número de vectores de estado arbitrarios estables en una memoria Hopfield es n.

Memoria Asociativa Bidireccional (BAM) de Kosko.

Bart Kosko, investigador de la *University of Southern California*, propuso en 1988 la *Bidireccional Associative Memory* (BAM) [14] para subsanar la clara desventaja de la autoasociatividad de la memoria Hopfield. La BAM maneja pares de vectores (A_1,B_1), ... (A_m,B_m), donde $A \in \{0, 1\}^n$ y $B \in \{0, 1\}^p$.

Al igual que Austin ensambló dos redes asociativas de Willshaw para diseñar su ADAM [33], Kosko ideó un arreglo de dos memorias Hopfield, y demostró que este diseño es capaz de asociar patrones de manera heteroasociativa.

La matriz **M** es una memoria Hopfield con la única diferencia que la diagonal principal es diferente de cero. $\mathbf{M}^T$ es la matriz transpuesta de **M** que, ahora como entrada, recibe a *B* y la salida será *A*. El proceso bidireccional anteriormente ilustrado continúa hasta que *A* y *B* convergen a una pareja estable (A_i, B_i).

$$
\begin{aligned}
&A \rightarrow \mathbf{M} \rightarrow B \\
&A' \leftarrow \mathbf{M}^T \leftarrow B \\
&A' \rightarrow \mathbf{M} \rightarrow B' \\
&A'' \leftarrow \mathbf{M}^T \leftarrow B' \\
&\qquad\qquad \bullet\bullet\bullet \\
&A_i \rightarrow \mathbf{M} \rightarrow B_i \\
&A_i \leftarrow \mathbf{M}^T \leftarrow B_i \\
&\cdots
\end{aligned}
$$

Kosko descubrió que su memoria funcionaba mejor con patrones bipolares que con patrones binarios (a la manera de Hopfield), por tanto: $A \in \{-1, 1\}^n$ y $B \in \{-1, 1\}^p$
Para la codificación de la BAM se superponen las *m* asociaciones sumándolas para formar la matriz de correlación:

$$\mathbf{M} = \sum_i A_i^T B_i \qquad \textbf{1.10}$$

y la memoria dual $\mathbf{M}^T$ que está dada por:

$$\mathbf{M}^T = \sum_i \left(A_i^T B_i\right)^t = \sum_i B_i^T A_i \qquad \textbf{1.11}$$

En el proceso de decodificación, cada neurona a_i que se encuentra en el campo *A* y cada neurona b_i localizada en el campo *B*, de forma independiente y asíncrona, examina la suma de entrada de las neuronas del otro campo, entonces puede o no cambiar su estado si la suma de entrada es mayor, igual o menor que un umbral dado. Si la suma de entrada es igual al umbral, entonces la neurona no cambia su estado. La suma de entrada para b_j es el producto interno columna:

$$A\mathbf{M}^j = \sum_i a_i m_{ij} \qquad \textbf{1.12}$$

donde $\mathbf{M}^j$ es la *j*-ésima columna de **M**. La suma de entrada para a_i es, de manera similar,

$$B\mathbf{M}_i^T = \sum_j b_j m_{ij} \qquad \textbf{1.13}$$

donde $\mathbf{M}_i$ es la *i*-ésima fila de **M**. Se toma el 0 como el umbral para todas las neuronas. Las funciones de umbral para a_i y b_j son:

$$a_i = \begin{cases} 1, & \text{si} \quad B\mathbf{M}_i^T > 0 \\ -1, & \text{si} \quad B\mathbf{M}_i^T < 0 \end{cases} \qquad \textbf{1.14}$$

$$b_j = \begin{cases} 1, & \text{si} \quad A\mathbf{M}^j > 0 \\ -1, & \text{si} \quad A\mathbf{M}^j < 0 \end{cases} \qquad \textbf{1.15}$$

Cuando se le presenta un patrón (*A*, *B*) a la BAM, las neuronas en los campos *A* y *B* se prenden o se apagan de acuerdo a la ocurrencia de 1's y 0's en los vectores de estado *A* y *B*. Las neuronas continúan sus cambios de estado hasta que se alcance un estado estable bidireccional (A_f, B_f).

Memorias Asociativas Morfológicas

La diferencia fundamental entre las memorias asociativas clásicas (*Lernmatrix*, *Correlograph*, *Linear Associator* y Memoria Asociativa Hopfield) y las memorias asociativas morfológicas radica en los fundamentos operacionales de éstas últimas, que son las operaciones morfológicas de *dilatación* y *erosión*; el nombre de las memorias asociativas morfológicas está inspirado precisamente en estas dos operaciones básicas.

Estas memorias rompieron con el esquema utilizado a través de los años en los modelos de memorias asociativas clásicas, que utilizan operaciones convencionales entre vectores y matrices para la fase de aprendizaje y suma de productos para la recuperación de patrones. Las memorias asociativas morfológicas cambian los productos por sumas y las sumas por máximos o mínimos en ambas fases, tanto de aprendizaje como de recuperación de patrones [19, 34, 35].

Hay dos tipos de memorias asociativas morfológicas: las memorias *max*, simbolizadas con **M**, y las memorias *min*, cuyo símbolo es **W**; en cada uno de los dos tipos, las memorias pueden funcionar en ambos modos: heteroasociativo y autoasociativo.

Se definen dos nuevos productos matriciales:

El *producto máximo* entre **D** y **H**, denotado por $\mathbf{C} = \mathbf{D} \nabla \mathbf{H}$, es una matriz $[c_{ij}]_{mxn}$ cuya *ij*-ésima componente c_{ij} es

$$c_{ij} = \bigvee_{k=1}^{r} \left(d_{ik} + h_{kj} \right) \qquad \textbf{1.16}$$

El *producto mínimo* de **D** y **H** denotado por $\mathbf{C} = \mathbf{D} \Delta \mathbf{H}$, es una matriz $[c_{ij}]_{mxn}$ cuya *ij*-ésima componente c_{ij} es

$$c_{ij} = \bigwedge_{k=1}^{r} \left(d_{ik} + h_{kj} \right) \qquad \textbf{1.17}$$

Los productos máximo y mínimo contienen a los operadores máximo y mínimo, los cuales están íntimamente ligados con los conceptos de las dos operaciones básicas de la morfología matemática: *dilatación* y *erosión*, respectivamente.

Memorias Heteroasociativas Morfológicas

Algoritmo de las memorias morfológicas max

Fase de Aprendizaje

1. Para cada una de las p asociaciones $(\mathbf{x}^{\mu}, \mathbf{y}^{\mu})$ se usa el producto mínimo para crear la matriz $\mathbf{y}^{\mu} \Delta (-\mathbf{x}^{\mu})^{t}$ de dimensiones m x n, donde el negado transpuesto del patrón de entrada $\mathbf{x}^{\mu}$ se define como $(-\mathbf{x}^{\mu})^{t} = \left(-x_1^{\mu}, -x_2^{\mu}, ..., x_n^{\mu}\right)$.
2. Se aplica el operador máximo $\vee$ a las p matrices para obtener la memoria **M**.

$$\mathbf{M} = \bigvee_{\mu=1}^{p} \left[\mathbf{y}^{\mu} \Delta (-\mathbf{x}^{\mu})^{t}\right] \qquad \textbf{1.18}$$

Fase de Recuperación

Esta fase consiste en realizar el producto mínimo Δ de la memoria **M** con el patrón de entrada $\mathbf{x}^{\omega}$, donde $\omega \in \{1, 2, ..., p\}$, para obtener un vector columna **y** de dimensión m:

$$\mathbf{y} = \mathbf{M} \Delta \mathbf{x}^{\omega} \qquad \textbf{1.19}$$

Las fases de aprendizaje y de recuperación de las **memorias morfológicas *min*** se obtienen por dualidad.

Memorias Autoasociativas Morfológicas

Para este tipo de memorias se utilizan los mismos algoritmos descritos anteriormente y que son aplicados a las memorias heteroasociativas; lo único que cambia es el conjunto fundamental. Para este caso, se considera el siguiente conjunto fundamental:

$$\{(\mathbf{x}^{\mu}, \mathbf{x}^{\mu}) \mid \mathbf{x}^{\mu} \in A^{n}, \text{ donde } \mu = 1, 2, ..., p\}$$

Memorias Asociativas Alfa-Beta

Las memorias asociativas Alfa-Beta [20] utilizan máximos y mínimos, y dos operaciones binarias originales α y β de las cuales heredan el nombre.

Para la definición de las operaciones binarias α y β se deben especificar los conjuntos A y B, los cuales son:

$$A = \{0, 1\} \qquad \text{y} \qquad B = \{0, 1, 2\}$$

La operación binaria α: A x $A \rightarrow B$ se define como:

x	y	$\alpha(x, y)$
0	0	1
0	1	0
1	0	2
1	1	1

La operación binaria β: B x $A \rightarrow A$ se define como:

x	y	$\beta(x, y)$
0	0	0
0	1	0
1	0	0
1	1	1
2	0	1
2	1	1

El fundamento teórico de las memorias asociativas Alfa-Beta se presenta en el siguiente capítulo de forma más completa, debido a que estas memorias son la base fundamental para este trabajo de tesis.

Memorias Asociativas Mediana

Las Memorias Asociativas Mediana [27] utilizan los operadores A y B, definidos de la siguiente forma:

$$A(x, y) = x - y \quad \textbf{1.20}$$

$$B(x, y) = x + y \quad \textbf{1.21}$$

Las operaciones utilizadas se describen a continuación.

Sean $P = [p_{ij}]_{mxr}$ y $Q=[q_{ij}]_{rxn}$ dos matrices.

Operación $\lozenge_A$: $P_{mxr} \lozenge_A Q_{rxn} = [f_{ij}^{A}]_{mxn}$ donde $f_{ij}^{A} = \underset{k=1}{\overset{r}{\mathbf{med}}} A(p_{ik}, q_{k,j})$ **1.22**

Operación $\lozenge_B$: $P_{mxr} \lozenge_B Q_{rxn} = [f_{ij}^{B}]_{mxn}$ donde $f_{ij}^{B} = \underset{k=1}{\overset{r}{\mathbf{med}}} B(p_{ik}, q_{k,j})$ **1.23**

Algoritmo de las Memorias Mediana

Fase de Aprendizaje

Paso 1. Para cada $\xi = 1, 2, ..., p$, de cada pareja $(\mathbf{x}^{\xi}, \mathbf{y}^{\xi})$ se construye la matriz:

$$[\mathbf{y}^{\xi} \lozenge_A (\mathbf{x}^{\xi})^t]_{mxn} \quad \textbf{1.24}$$

Paso 2. Se aplica el operador media a las matrices obtenidas en le paso 1 para obtener la matriz **M**, como sigue:

$$\mathbf{M} = \underset{\xi=1}{\overset{p}{\mathbf{med}}} \left[y^{\xi} \lozenge_A \left(x^{\xi} \right)^t \right] \quad \textbf{1.25}$$

El ij-ésimo componente **M** está dado como sigue:

$$m_{ij} = \underset{\xi=1}{\overset{p}{\mathbf{med}}} \mathrm{A}\left(y_i^{\xi}, x_j^{\xi}\right) \quad \textbf{1.26}$$

Fase de Recuperación

Se tienen dos casos:

Caso 1. Recuperación de un patrón fundamental. Un patrón $\mathbf{x}^w$, con $w \in \{1, 2, ..., p\}$ se le presenta a la memoria **M** y se realiza la siguiente operación:

$$\mathbf{M} \lozenge_{\mathrm{B}} \mathbf{x}^w \quad \textbf{1.27}$$

El resultado es un vector columna de dimensión *n*, con la *i*-ésima componente dada como:

$$\left(\mathbf{M} \lozenge_B x^w\right)_i = \underset{j=1}{\overset{n}{\mathbf{med}}} \mathrm{B}\left(m_{ij}, x_j^w\right) \quad \textbf{1.28}$$

Caso 2. Recuperación de un patrón alterado. Un patrón $\tilde{\mathbf{x}}$, que es una versión alterada de un patrón $\mathbf{x}^w$, se le presenta a la memoria **M** y se realiza la siguiente operación:

$$\mathbf{M} \lozenge_{\mathrm{B}} \tilde{\mathbf{x}} \quad \textbf{1.29}$$

De nuevo, es resultado es un vector de dimensión *n*, con la *i*-ésima componente dada como:

$$\left(\mathbf{M} \lozenge_B \tilde{\mathbf{x}}\right)_i = \underset{j=1}{\overset{n}{\mathbf{med}}} \mathrm{B}\left(m_{ij}, \tilde{x}_j\right) \quad \textbf{1.30}$$

CAPÍTULO 3. Materiales y Métodos

ste capítulo contiene dos secciones. En la sección 3.1 trataremos los conceptos básicos sobre bioinformática que son utilizados en esta tesis. En la sección 3.2 se describe a detalle el modelo matemático de las Memorias Asociativas Alfa-Beta, el cual es la base fundamental del algoritmo que da lugar al nuevo modelo de memoria asociativa propuesto en este trabajo de tesis.

1.1 Bioinformática

En la presente sección se exponen dos de los principales problemas que se intentan resolver en bioinformática y que son tratados en esta tesis, *promotores y zonas de empalme (splice-junction)*. Se definirán cada uno de estos temas y se expondrá cuál es la importancia de los mismos.

1.1.1 Promotores

Los mecanismos que regulan la *expresión genética* de los organismos, tanto eucariontes como procariontes, son los mismos que permiten la adaptación de los seres vivos a su medio ambiente. Los elementos principales en dicha adaptación son: el ADN, RNA y las proteínas [71-73]. Una de las partes más importantes dentro de la expresión genética es la llamada *transcripción*, que es el proceso de síntesis, en el cual se codifica una secuencia de ADN en una secuencia mRNA, la cual será el molde para la creación de una proteína.

La *fase inicial de la transcripción* es fundamental, ya que es el comienzo de todo el proceso de síntesis de una proteína. En este proceso el RNA polimerasa se adhiere a un segmento de secuencia particular del ADN, al que se denomina el *promotor*, y a partir de aquí se comienza el proceso de síntesis, localizando los segmentos específicos que codificarán para una proteína en específico. Los promotores están localizados en las regiones previas, en el sentido inverso en el que se lleva a cabo la transcripción (upstream), a la de los genes transcritos [73]. La frecuencia con la que se inicia la transcripción depende de la afinidad del RNA polimerasa por el promotor; se sabe que esta afinidad se puede incrementar hasta 100 veces dependiendo de los bp (pares bases, por sus siglas en ingles) o secuencias de nucleótidos que conforman al promotor [74,75].

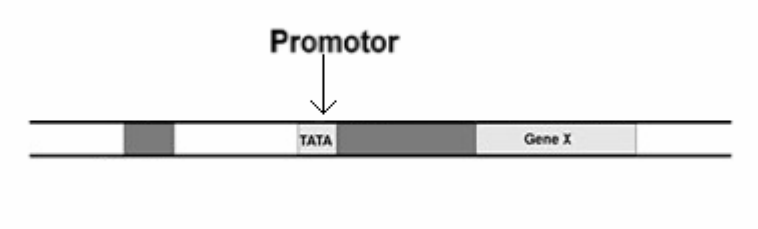


Figura 1.1 Promotor**.**

Promotores Eucariontes

Los promotores en los eucariontes (seres vivos cuyas células contienen membrana celular definida) consisten de una secuencia de elementos que es posible localizar, usualmente, en el sentido *upstream* a partir del sitio del comienzo de la transcripción, y es reconocido por los factores de transcripción. Ambos, tanto elementos en el sitio del comienzo de la transcripción como los factores de transcripción, es común que aparezcan en una gran variedad de promotores y son usadas constantemente, mientras que otros elementos y sus respectivos factores son específicos a cierta clase de genes [76]. Además suele suceder que los promotores se combinen por lo que es necesario el uso de los distintos factores de transcripción para su iniciación. Por lo tanto, en la iniciación del proceso de transcripción en los promotores de las seres eucariontes, es necesario el uso de diversos factores de transcripción por lo que en lugar de una sola RNA polimerasa existen 3: *RNA polimerasa I, RNA polimerasa II, RNA polimerasa III*, estos tres tipos de factores de transcripción se un a varios tipos de promotores, dependiendo sus características. El RNA polimerasa I y II se unen principalmente con sitios de transcripción que tienen un sentido *upstream*. Algunos de los promotores a los que se unen el RNA polimerasa III estás principalmente en sentido *downstream*. La RNA polimerasa I y III reconocen un conjunto restringido de promotores y que dependen de un número reducido de *factores accesorios* (proteínas que ayudan con la transcripción pero que no hacen contacto directo con el factor de transcripción fundamental) [77-79].

Promotores Procariontes (Escherichia Coli)

Más de 300 promotores han sido caracterizados experimentalmente por varios investigadores. Existen cuatro características principales en la mayoría de los promotores de E. Coli: *sitio de comienzo de la transcripción, el hexámero -10, el hexámero -35,y la distancia entre las secuencias -10 y -35.* En el caso del sitio de comienzo de la transcripción se ha visto que en más de un 90% se encuentra un purina (Adenina o Guanina) [80]. La segunda característica es una cadena de seis pares bases, TATAAT, reconocible en la mayoría de los promotores y se encuentra aproximadamente a una distancia de 10 pares bases del sitio de comienzo de la transcripción. La tercera característica es el es un hexámero que se encuentra aproximadamente a 35 pares base de sitio del comienzo de la transcripción, y el consenso de dicho hexámero es TTGACA [80]. Y por último la distancia entre estos dos hexámeros es considerada también como factor fundamental en la localización de promotores en E. Coli, se ha encontrado que la distancia aproximada que los separa es de 16 a 18 pares bases en el 90% de los promotores [80].

1.1.2 Localización de zonas de empalme (Splice-Junction Location)

Uno de los descubrimientos más importantes en lo que respecta a los organismos eucariontes fue hecho en 1977, cuando se llegó a la conclusión de que los genes contenían secuencias extras que no aparecían en el mRNA. Estas secuencias son conocidas como intrones, mientras que las secuencias que sí codifican son llamadas exones. Cuando se lleva a cabo la codificación del mRNA es necesario separar los intrones de los exones para así poder obtener una secuencia continua para la translación. Los intrones varían en cantidad y tamaño, y en ocasiones son más grandes que los exones. Por ejemplo, dos genes del mismo tamaño pueden tener un número diferente de intrones y, además, variar éstos en tamaño [85].

Las zonas de cambio entre un exon-intron (EI) o intron-exon (IE) son conocidos como *sitios de empalme (splice-sites)*. Dichas zonas son consideradas *superfluas* ya que en el proceso de codificación de una proteína es necesario remover estas zonas (intrones) para poder obtener la secuencia que codifique para una proteína especifica. [49-50].

El separar de manera correcta los intrones de los exones, es una tarea muy importante ya que es parte de un mecanismo conocido como *expresión genética*. Una gran cantidad de genes pueden dar lugar a más de un tipo de mRNA codificado con el objeto de producir variantes proteínicas. A este proceso se le conoce como *empalme alternativo*. Un ejemplo de éste se puede ver en la fig. 3.2 que muestra cómo, a partir de una secuencia genética con tres exones separado por sus respectivos intrones, es posible obtener dos genes (mRNA1 y mRNA2) que darán paso a dos proteínas distintas [85,86].

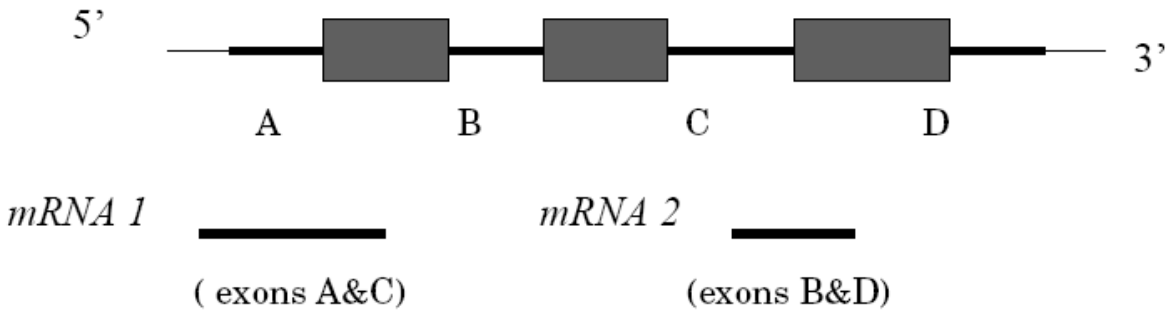


Figura 1.2 Empalme Alternativo**.**

La alteración de este proceso puede generar serios problemas, ya que el no identificar correctamente los sitios de empalme, dará como resultado la codificación de una proteína no deseada, con lo que algún proceso biológico fallará, causando problemas en el correspondiente ser vivo. A este tipo de problema se le conoce como *mutación de sitios de empalme* [85,86].

Por último, cabe mencionar que la identificación de los genes y regiones de codificación en secuencias de DNA es una de las metas principales en la biología molecular, por lo que desarrollar algoritmos computacionales que permitan realizar esta tarea, se convierte en uno de los problemas fundamentales en Bioinformática [86].

1.2 Memorias Asociativas Alfa-Beta

En la presente sección se expone el modelo fundamental sobre el que se basa este trabajo de tesis, las *Memorias Asociativas Alfa-Beta*. Presentando las definiciones de las operaciones α y β, las operaciones matriciales utilizando estas operaciones originales, y se describe la fase de aprendizaje y recuperación de las memorias heteroasociativas y autoasociativas Alfa-Beta, tanto $\vee$ (*max*) como $\wedge$ (*min*).

La numeración de los Lemas y Teoremas que aparecen aquí, es respetada tal y como se presentan en [20].

1.2.1 Operaciones binarias α y β: definiciones y propiedades

Las memorias Alfa-Beta utilizan máximos y mínimos, y dos operaciones binarias originales α y β de las cuales heredan el nombre.

Para la definición de las operaciones binarias α y β se deben especificar los conjuntos A y B, los cuales son:

$$A = \{0, 1\} \quad \text{y} \quad B = \{0, 1, 2\}$$

La operación binaria $\alpha: A \times A \rightarrow B$ se define como se muestra en la Tabla 1.1.

Tabla 1.1 Operación binaria $\alpha: A \times A \rightarrow B$

x	Y	$\alpha(x, y)$
0	0	1
0	1	0
1	0	2
1	1	1

La operación binaria $\beta: B \times A \rightarrow A$ se define como se muestra en la Tabla 1.2.

Tabla 1.2 Operación binaria $\beta: B \times A \rightarrow A$

x	Y	$\beta(x, y)$
0	0	0
0	1	0
1	0	0
1	1	1
2	0	1
2	1	1

Los conjuntos *A* y *B*, las operaciones binarias α y β junto con los operadores $\wedge$ (mínimo) y $\vee$ (máximo) usuales conforman el sistema algebraico (A, B, α, β, $\wedge$, $\vee$) en el que están inmersas las memorias asociativas Alfa-Beta [20, 37].

Se requiere la definición de cuatro operaciones matriciales, de las cuales se usarán sólo 4 casos particulares:

Operación ***α max***: $P_{mxr} \nabla_{\alpha} Q_{rxn} = \left[f_{ij}^{\alpha} \right]_{mxn}$, donde $f_{ij}^{\alpha} = \bigvee_{k=1}^{r} \alpha(p_{ik}, q_{kj})$ **1.1**

Operación ***β max***: $P_{mxr} \nabla_{\beta} Q_{rxn} = \left[f_{ij}^{\beta} \right]_{mxn}$, donde $f_{ij}^{\beta} = \bigvee_{k=1}^{r} \beta(p_{ik}, q_{kj})$ **1.2**

Operación ***α min***: $P_{mxr} \Delta_{\alpha} Q_{rxn} = \left[h_{ij}^{\alpha} \right]_{mxn}$, donde $h_{ij}^{\alpha} = \bigwedge_{k=1}^{r} \alpha(p_{ik}, q_{kj})$ **1.3**

Operación ***β min***: $P_{mxr} \Delta_{\beta} Q_{rxn} = \left[h_{ij}^{\beta} \right]_{mxn}$, donde $h_{ij}^{\beta} = \bigwedge_{k=1}^{r} \beta(p_{ik}, q_{kj})$ **1.4**

El siguiente lema muestra los resultados obtenidos al utilizar las operaciones que involucran al operador binario α con las componentes de un vector columna y un vector fila dados.

Lema 3.9. (*Numeración tal como aparece en* [20]). *Sean* $\mathbf{x} \in A^n$ *y* $\mathbf{y} \in A^m$*; entonces* $\mathbf{y} \nabla_{\alpha} \mathbf{x}^t$ *es una matriz de dimensiones m x n, y además se cumple que:* $\mathbf{y} \nabla_{\alpha} \mathbf{x}^t = \mathbf{y} \Delta_{\alpha} \mathbf{x}^t$.

Demostración.

$$\mathbf{y} \nabla_{\alpha} \mathbf{x}^t = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix} \nabla_{\alpha} \left(x_1, x_2, \ldots, \mathrm{x_n} \right)$$

$$= \begin{pmatrix} \bigvee_{k=1}^{1} \alpha(y_1, x_1) & \bigvee_{k=1}^{1} \alpha(y_1, x_2) & \cdots & \bigvee_{k=1}^{1} \alpha(y_1, x_n) \\ \bigvee_{k=1}^{1} \alpha(y_2, x_1) & \bigvee_{k=1}^{1} \alpha(y_2, x_2) & \cdots & \bigvee_{k=1}^{1} \alpha(y_2, x_n) \\ \vdots & \vdots & \cdots & \vdots \\ \bigvee_{k=1}^{1} \alpha(y_m, x_1) & \bigvee_{k=1}^{1} \alpha(y_m, x_2) & \cdots & \bigvee_{k=1}^{1} \alpha(y_m, x_n) \end{pmatrix}_{mxn}$$

$$= \begin{pmatrix} \alpha(y_1, x_1) & \alpha(y_1, x_2) & \cdots & \alpha(y_1, x_n) \\ \alpha(y_2, x_1) & \alpha(y_2, x_2) & \cdots & \alpha(y_2, x_n) \\ \vdots & \vdots & \cdots & \vdots \\ \alpha(y_m, x_1) & \alpha(y_m, x_2) & \cdots & \alpha(y_m, x_n) \end{pmatrix}_{mxn}$$

$$= \begin{pmatrix} \bigwedge_{k=1}^{1} \alpha(y_1, x_1) & \bigwedge_{k=1}^{1} \alpha(y_1, x_2) & \cdots & \bigwedge_{k=1}^{1} \alpha(y_1, x_n) \\ \bigwedge_{k=1}^{1} \alpha(y_2, x_1) & \bigwedge_{k=1}^{1} \alpha(y_2, x_2) & \cdots & \bigwedge_{k=1}^{1} \alpha(y_2, x_n) \\ \vdots & \vdots & \cdots & \vdots \\ \bigwedge_{k=1}^{1} \alpha(y_m, x_1) & \bigwedge_{k=1}^{1} \alpha(y_m, x_2) & \cdots & \bigwedge_{k=1}^{1} \alpha(y_m, x_n) \end{pmatrix}_{mxn}$$

$$= \mathbf{y} \Delta_{\alpha} \mathbf{x}^{t}$$

En efecto, resulta que $\mathbf{y} \nabla_{\alpha} \mathbf{x}^{t}$ es una matriz de dimensiones *m* x *n*, y que $\mathbf{y} \nabla_{\alpha} \mathbf{x}^{t} = \mathbf{y} \Delta_{\alpha} \mathbf{x}^{t}$.

Dado el resultado del lema anterior, es conveniente escoger un símbolo único, digamos el símbolo $\boxtimes$, que represente a las dos operaciones ∇_{α} y Δ_{α} cuando se opera un vector columna de dimensión *m* con un vector fila de dimensión *n*:

$$\mathbf{y} \nabla_{\alpha} \mathbf{x}^{t} = \mathbf{y} \boxtimes \mathbf{x}^{t} = \mathbf{y} \Delta_{\alpha} \mathbf{x}^{t} \qquad \textbf{1.5}$$

La *ij*-ésima componente de la matriz está $\mathbf{y} \boxtimes \mathbf{x}^{t}$ dada por:

$$[\mathbf{y} \boxtimes \mathbf{x}^{t}]_{ij} = \alpha(y_i, x_j) \qquad \textbf{1.6}$$

Dado un índice de asociación μ, la expresión anterior indica que la *ij*-ésima componente de la matriz $\mathbf{y}^{\mu} \boxtimes (\mathbf{x}^{\mu})^{t}$ se expresa de la siguiente manera:

$$[\mathbf{y}^{\mu} \boxtimes (\mathbf{x}^{\mu})^{t}]_{ij} = \alpha\left(y_i^{\mu}, x_j^{\mu}\right) \qquad \textbf{1.7}$$

Ahora se analizará el caso en el que se opera una matriz de dimensiones *m* x *n* con un vector columna de dimensión *n* usando las operaciones∇_{β} y Δ_{β}. En los lemas 3.11 y 3.12 se obtiene la forma que exhibirán las *i*-ésimas componentes de los vectores columna resultantes de dimensión *m*, a partir de ambas operaciones ∇_{β} y Δ_{β}.

Lema 3.11. (*Numeración tal como aparece en* [20]). *Sean* $\mathbf{x} \in A^{n}$ y $\mathbf{P}$ *una matriz de dimensiones m x n. La operación* $\mathbf{P}_{mxn} \nabla_{\beta} \mathbf{x}$ *da como resultado un vector columna de dimensión m, cuya i-ésima componente tiene la siguiente forma:*

$$\left(\mathbf{P}_{mxn} \nabla_{\beta} \mathbf{x}\right)_i = \bigvee_{j=1}^{n} \beta\left(p_{ij}, x_j\right) \qquad \textbf{1.8}$$

Demostración.

$$\mathbf{P}_{mxn} \nabla_{\beta} \mathbf{x} = \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & p_{22} & \cdots & p_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ p_{m1} & p_{m2} & \cdots & p_{mn} \end{pmatrix} \nabla_{\beta} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$

$$\mathbf{P}_{mxn}\nabla_{\beta}\mathbf{x} = \begin{pmatrix} \beta(p_{11},x_1)\vee\beta(p_{12},x_2)\vee\ldots\vee\beta(p_{1n},x_n) \\ \beta(p_{21},x_1)\vee\beta(p_{22},x_2)\vee\ldots\vee\beta(p_{2n},x_n) \\ \vdots \\ \beta(p_{m1},x_1)\vee\beta(p_{m2},x_2)\vee\ldots\vee\beta(p_{mn},x_n) \end{pmatrix} = \begin{pmatrix} \bigvee_{j=1}^{n}\beta(p_{1j},x_j) \\ \bigvee_{j=1}^{n}\beta(p_{2j},x_j) \\ \vdots \\ \bigvee_{j=1}^{n}\beta(p_{mj},x_j) \end{pmatrix}$$

Se obtiene un vector columna de dimensión *m* cuya *i*-ésima componente es

$$\left(\mathbf{P}_{mxn}\nabla_{\beta}\mathbf{x}\right)_i = \bigvee_{j=1}^{n}\beta\left(p_{ij},x_j\right) \qquad \textbf{1.9}$$

Lema 3.12. (*Numeración tal como aparece en* [20]). *Sean* $\mathbf{x} \in A^n$ y $\mathbf{P}$ *una matriz de dimensiones m x n. La operación* $\mathbf{P}_{mxn}\ \Delta_{\beta}\ \mathbf{x}$ *da como resultado un vector columna de dimensión m, cuya i-ésima componente tiene la siguiente forma:* $\left(\mathbf{P}_{mxn}\Delta_{\beta}\mathbf{x}\right)_i = \bigwedge_{j=1}^{n}\beta\left(p_{ij},x_j\right)$

Demostración.

$$\mathbf{P}_{mxn}\Delta_{\beta}\mathbf{x} = \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & p_{22} & \cdots & p_{2n} \\ \vdots & \vdots & \ldots & \vdots \\ p_{m1} & p_{m2} & \cdots & p_{mn} \end{pmatrix}\Delta_{\beta}\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$

$$\mathbf{P}_{mxn}\Delta_{\beta}\mathbf{x} = \begin{pmatrix} \beta(p_{11},x_1)\wedge\beta(p_{12},x_2)\wedge\ldots\wedge\beta(p_{1n},x_n) \\ \beta(p_{21},x_1)\wedge\beta(p_{22},x_2)\wedge\ldots\wedge\beta(p_{2n},x_n) \\ \vdots \\ \beta(p_{m1},x_1)\wedge\beta(p_{m2},x_2)\wedge\ldots\wedge\beta(p_{mn},x_n) \end{pmatrix} = \begin{pmatrix} \bigwedge_{j=1}^{n}\beta(p_{1j},x_j) \\ \bigwedge_{j=1}^{n}\beta(p_{2j},x_j) \\ \vdots \\ \bigwedge_{j=1}^{n}\beta(p_{mj},x_j) \end{pmatrix}$$

Se obtiene un vector columna de dimensión *m* cuya *i*-ésima componente es

$$\left(\mathbf{P}_{mxn}\Delta_{\beta}\mathbf{x}\right)_i = \bigwedge_{j=1}^{n}\beta\left(p_{ij},x_j\right) \qquad \textbf{1.10}$$

1.2.2 Memorias Heteroasociativas Alfa-Beta

Se tienen dos tipos de memorias heteroasociativas Alfa-Beta: tipo **V** y tipo **Λ**. En la generación de ambos tipos de memorias se usará el operador $\boxtimes$ el cual tiene la siguiente forma:

$$[\mathbf{y}^{\mu}\boxtimes(\mathbf{x}^{\mu})^{t}]_{ij} = \alpha(y_i^{\mu},x_j^{\mu});\mu\in\{1,2,\ldots p\},i\in\{1,2,\ldots,p\},j\in\{1,2,\ldots n\} \qquad \textbf{1.11}$$

Algoritmo Memorias Alfa-Beta tipo V

Fase de Aprendizaje

Paso 1. Para cada $\mu = 1, 2, ..., p$, a partir de la pareja $(\mathbf{x}^\mu, \mathbf{y}^\mu)$ se construye la matriz

$$[\mathbf{y}^\mu \boxtimes (\mathbf{x}^\mu)^t]_{mxn} \qquad \textbf{1.12}$$

Paso 2. Se aplica el operador binario máximo $\vee$ a las matrices obtenidas en el paso 1:

$$\mathbf{V} = \bigvee_{\mu=1}^{p} [\mathbf{y}^\mu \boxtimes (\mathbf{x}^\mu)^t] \qquad \textbf{1.13}$$

La entrada *ij*-ésima está dada por la siguiente expresión:

$$v_{ij} = \bigvee_{\mu=1}^{p} \alpha(y_i^\mu, x_j^\mu) \qquad \textbf{1.14}$$

Fase de Recuperación

Se presenta un patrón $\mathbf{x}^\omega$, con $\omega \in \{1, 2, ..., p\}$, a la memoria heteroasociativa $\alpha\beta$ tipo **V** y se realiza la operación Δ_β: $\mathbf{V}\ \Delta_\beta\ \mathbf{x}^\omega$.

Dado que las dimensiones de la matriz **V** son de *m* x *n* y $\mathbf{x}^\omega$ es un vector columna de dimensión *n*, el resultado de la operación anterior debe ser un vector columna de dimensión *m*, cuya *i*-ésima componente es:

$$(\mathbf{V}\Delta_\beta x^\varpi)_i = \bigwedge_{j=1}^{n} \beta(v_{ij}, x_j^\varpi) \qquad \textbf{1.15}$$

Teorema 4.7. (*Numeración tal como aparece en* [20]). *Sea* $\{(\mathbf{x}^\mu, \mathbf{y}^\mu) \mid \mu = 1, 2, \ldots, p\}$ *el conjunto fundamental de una memoria heteroasociativa Alfa-Beta representada por* $\mathbf{V}$. *Si* ω *es un valor de índice arbitrario tal que* $\omega \in \{1, 2, \ldots, p\}$, *y si además para cada* $i \in \{1, \ldots, m\}$ *se cumple que* $\exists j = j_0 \in \{1, \ldots, n\}$, *el cual depende de* ω *y de* i, *tal que* $v_{ij_0} = \alpha(y_i^\omega, x_{j_0}^\omega)$, *entonces la recuperación* $\mathbf{V}\Delta_\beta\mathbf{x}^\omega$ *es correcta; es decir* $\mathbf{V}\Delta_\beta\mathbf{x}^\omega = \mathbf{y}^\omega$.

Demostración. Ver detalles de la demostración en [20].

Algoritmo Memorias Alfa-Beta tipo Λ

Fase de Aprendizaje

Paso 1. Para cada $\mu = 1, 2, ..., p$, a partir de la pareja $(\mathbf{x}^\mu, \mathbf{y}^\mu)$ se construye la matriz

$$[\mathbf{y}^{\mu} \boxtimes (\mathbf{x}^{\mu})^{t}]_{mxn} \quad \textbf{1.16}$$

Paso 2. Se aplica el operador binario mínimo $\wedge$ a las matrices obtenidas en el paso 1:

$$\mathbf{\Lambda} = \bigwedge_{\mu=1}^{p} [\mathbf{y}^{\mu} \boxtimes (\mathbf{x}^{\mu})^{t}] \quad \textbf{1.17}$$

La entrada *ij*-ésima está dada por la siguiente expresión:

$$\lambda_{ij} = \bigwedge_{\mu=1}^{p} \alpha(y_i^{\mu}, x_j^{\mu}) \quad \textbf{1.18}$$

Fase de Recuperación

Se presenta un patrón $\mathbf{x}^{\omega}$, con $\omega \in \{1, 2, ..., p\}$, a la memoria heteroasociativa *Alfa-Beta* tipo $\mathbf{\Lambda}$ y se realiza la operación ∇_{β}: $\mathbf{\Lambda} \nabla_{\beta} \mathbf{x}^{\omega}$.

Dado que las dimensiones de la matriz $\mathbf{\Lambda}$ son de *m* x *n* y $\mathbf{x}^{\omega}$ es un vector columna de dimensión *n*, el resultado de la operación anterior debe ser un vector columna de dimensión *m*, cuya *i*-ésima componente es:

$$\left(\mathbf{\Lambda} \nabla_{\beta} \mathbf{x}^{\omega}\right)_i = \bigvee_{j=1}^{n} \beta\left(\lambda_{ij}, x_j^{\mu}\right) \quad \textbf{1.19}$$

Teorema 4.20. (*Numeración tal como aparece en* [20]). *Sea* $\{(\mathbf{x}^{\mu}, \mathbf{y}^{\mu}) \mid \mu = 1, 2, \ldots, p\}$ *el conjunto fundamental de una memoria heteroasociativa Alfa-Beta representada por* $\mathbf{\Lambda}$ *. Si* ω *es un valor arbitrario fijo tal que* $\omega \in \{1, 2, \ldots, p\}$ *, y si además para cada* $i \in \{1, \ldots, m\}$ *se cumple que* $\exists j = j_0 \in \{1, \ldots, n\}$ *, el cual depende de* ω *y de* i *, tal que* $\lambda_{ij_0} = \alpha(y_i^{\omega}, x_{j_0}^{\omega})$ *, entonces la recuperación* $\mathbf{\Lambda} \nabla_{\beta} \mathbf{x}^{\omega}$ *es correcta; es decir* $\mathbf{\Lambda} \nabla_{\beta} \mathbf{x}^{\omega} = \mathbf{y}^{\omega}$.

Demostración. Ver detalles de la demostración en [20].

1.2.3 Memorias Autoasociativas Alfa-Beta

Si a una memoria heteroasociativa se le impone la condición de que $\mathbf{y}^{\mu} = \mathbf{x}^{\mu}$ $\forall \mu \in \{1, 2, ..., p\}$ entonces, deja de ser heteroasociativa y ahora se le denomina autoasociativa.

A continuación se enlistan algunas de las características de las memorias autoasociativas Alfa-Beta:

1. El conjunto fundamental toma la forma $\{(\mathbf{x}^{\mu}, \mathbf{x}^{\mu}) \mid \mu = 1, 2, ..., p\}$

2. Los patrones fundamentales de entrada y salida son de la misma dimensión; denotémosla por *n*.
3. La memoria es una matriz cuadrada, para ambos tipos, $\mathbf{V}$ y $\mathbf{\Lambda}$. Si $\mathbf{x}^{\mu} \in A^n$ entonces

$$\mathbf{V} = \left[v_{ij}\right]_{nxn} \text{ y } \Lambda = \left[\lambda_{ij}\right]_{nxn} \quad \textbf{1.20}$$

Algoritmo Memorias Autoasociativas Alfa-Beta tipo V

Las fases de aprendizaje y recuperación son similares a las memorias heteroasociativas Alfa-Beta.

Fase de Aprendizaje

Paso 1. Para cada $\mu = 1, 2, ..., p$, a partir de la pareja $(\mathbf{x}^{\mu},\mathbf{x}^{\mu})$ se construye la matriz

$$[\mathbf{x}^{\mu} \boxtimes (\mathbf{x}^{\mu})^{t}]_{mxn} \quad \textbf{1.21}$$

Paso 2. Se aplica el operador binario máximo $\mathbf{V}$ a las matrices obtenidas en el paso 1:

$$\mathbf{V} = \bigvee_{\mu=1}^{p} [\mathbf{x}^{\mu} \boxtimes (\mathbf{x}^{\mu})^{t}] \quad \textbf{1.22}$$

La entrada *ij*-ésima de la memoria está dada así:

$$v_{ij} = \bigvee_{\mu=1}^{p} \alpha(x_i^{\mu}, x_j^{\mu}) \quad \textbf{1.23}$$

y de acuerdo con que α: $A \times A \rightarrow B$, se tiene que $v_{ij} \in B$, $\forall i \in \{1, 2, ..., n\}$. $\forall j \in \{1, 2, ..., n\}$.

Fase de Recuperación

La fase de recuperación de las memorias autoasociativas Alfa-Beta tipo V tiene dos casos posibles. En el primer caso el patrón de entrada es un patrón fundamental; es decir, la entrada es un patrón $\mathbf{x}^{\omega}$, con $\omega \in \{1, 2, ..., p\}$. En el segundo caso, el patrón de entrada NO es un patrón fundamental, sino la versión distorsionada de por lo menos uno de los patrones fundamentales; lo anterior significa que si el patrón de entrada es $\tilde{\mathbf{x}}$, debe existir al menos un valor de índice $\omega \in \{1, 2, ..., p\}$, que corresponde al patrón fundamental respecto del cual $\tilde{\mathbf{x}}$ es una versión alterada.

CASO 1: Patrón fundamental. Se presenta a un patrón $\mathbf{x}^{\omega}$, con $\omega \in \{1, 2, ..., p\}$ a la memoria autoasociativa Alfa-Beta tipo $\mathbf{V}$ y se realiza la operación Δ_{β}:

$$\mathbf{V}\ \Delta_{\beta}\ \mathbf{x}^{\omega} \quad \textbf{1.24}$$

El resultado de la operación anterior será el vector columna de dimensión *n*.

$$\left(\mathbf{V}\Delta_{\beta}\mathbf{x}^{\omega}\right)_i = \bigwedge_{j=1}^{n} \beta(v_{ij}, x_j^{\omega}) \qquad \textbf{1.25}$$

$$\left(\mathbf{V}\Delta_{\beta}\mathbf{x}^{\omega}\right)_i = \bigwedge_{j=1}^{n} \beta\left\{\left[\bigvee_{\mu=1}^{p} \alpha(x_i^{\mu}, x_j^{\mu})\right], x_j^{\omega}\right\} \qquad \textbf{1.26}$$

CASO 2: Patrón alterado. Se presenta el patrón binario $\tilde{\mathbf{x}}$ (patrón alterado de algún patrón fundamental $\mathbf{x}^{\omega}$) que es un vector columna de dimensión *n*, a la memoria autoasociativa Alfa-Beta tipo V y se realiza la operación

$$\mathbf{V}\Delta_{\beta}\tilde{\mathbf{x}} \qquad \textbf{1.27}$$

Al igual que en el caso 1, el resultado de la operación anterior es un vector columna de dimensión *n*, cuya *i*-ésima componente se expresa de la siguiente manera:

$$\left(\mathbf{V}\Delta_{\beta}\tilde{\mathbf{x}}\right)_i = \bigwedge_{j=1}^{n} \beta(v_{ij}, \tilde{x}_j) \qquad \textbf{1.28}$$

$$\left(\mathbf{V}\Delta_{\beta}\tilde{\mathbf{x}}\right)_i = \bigwedge_{j=1}^{n} \beta\left\{\left[\bigvee_{\mu=1}^{p} \alpha(x_i^{\mu}, x_j^{\mu})\right], \tilde{x}_j\right\} \qquad \textbf{1.29}$$

Lema 4.27. (*Numeración tal como aparece en* [20]). *Una memoria autoasociativa Alfa-Beta tipo* **V** *tiene únicamente unos en la diagonal principal*.

Demostración. La *ij*-ésima entrada de una memoria autoasociativa Alfa-Beta tipo **V** está dada por $v_{ij} = \bigvee_{\mu=1}^{p} \alpha\left(x_i^{\mu}, x_j^{\mu}\right)$. Las entradas de la diagonal principal se obtienen de la expresión anterior haciendo $i = j$:

$$v_{ii} = \bigvee_{\mu=1}^{p} \alpha\left(x_i^{\mu}, x_i^{\mu}\right), \quad \forall i \in \{1,2,...n\} \qquad \textbf{1.30}$$

Por la propiedad de la tabla se tiene que $\alpha(x_i^{\mu}, x_i^{\mu}) = 1$, por lo que la expresión anterior se transforma en:

$$v_{ii} = \bigvee_{\mu=1}^{p} (1) = 1, \quad \forall i \in \{1,2,...n\} \qquad \textbf{1.31}$$

Teorema 4.28. (*Numeración tal como aparece en* [20]). *Una memoria autoasociativa Alfa-Beta tipo* **V** *recupera de manera correcta el conjunto fundamental completo; además, tiene máxima capacidad de aprendizaje.*

Demostración. Sea $\omega = \{1, 2, ..., p\}$ arbitrario. De acuerdo con el lema 4.27, para cada $i \in \{1, ..., n\}$ escogida arbitrariamente

$$v_{ii} = 1 = \alpha(x_i^{\omega}, x_i^{\omega}) \qquad \textbf{1.32}$$

Es decir, para $i \in \{1, ..., n\}$ escogida arbitrariamente, $\exists\, j_0 = i \in \{1, ..., n\}$ que cumple con:

$$v_{ij_0} = \alpha(x_i^{\omega}, x_{j_0}^{\omega}) \qquad \textbf{1.33}$$

Por lo tanto, de acuerdo con el Teorema 4.7

$$\mathbf{V}\, \Delta_{\beta}\, \mathbf{x}^{\omega} = \mathbf{x}^{\omega},\ \forall \omega \in \{1, 2, ..., p\} \qquad \textbf{1.34}$$

Esto significa que la memoria autoasociativa Alfa-Beta tipo V recupera de manera correcta el conjunto fundamental completo.

Además, en la demostración de este Teorema, en ningún momento aparece restricción alguna sobre *p*, que es la cardinalidad del conjunto fundamental; y esto quiere decir que el conjunto fundamental puede crecer tanto como se quiera. La consecuencia directa es que el número de patrones que puede aprender una memoria autoasociativa Alfa-Beta tipo V, con recuperación correcta, es máximo.

El Teorema 4.28 se puede enunciar desde un enfoque matricial de la siguiente manera: dado que para cada asociación $(\mathbf{x}^{\omega},\mathbf{x}^{\omega})$ del conjunto fundamental de una memoria autoasociativa Alfa-Beta V, se cumple que cada fila de la matriz V - $\mathbf{x}^{\omega} \boxtimes (\mathbf{x}^{\omega})^t$ contiene una entrada cero, entonces de la memoria V recupera el conjunto completo de patrones fundamentales en forma correcta.

Teorema 4.30. (*Numeración tal como aparece en* [20]). *Sea* $\{(\mathbf{x}^{\mu},\mathbf{x}^{\mu}) \mid \mu = 1, 2, ..., p\}$ *el conjunto fundamental de una memoria autoasociativa Alfa-Beta representada por* **V**, *y sea* $\tilde{\mathbf{x}} \in A^n$ *un patrón alterado positivamente respecto a algún patrón fundamental* $\mathbf{x}^{\omega}$ con $\omega \in \{1, 2, ..., p\}$. *Si se presenta* $\tilde{\mathbf{x}}$ *a la memoria* **V** *como entrada, y si además para cada* $i \in \{1, ..., n\}$ *se cumple la condición de que* $\exists j = j_0 \in \{1, ..., n\}$, *el cual depende de* ω *y de i tal* que $v_{ij_0} \le \alpha(x^{\omega}, \tilde{x}_{j_0})$, *entonces la recuperación* $\mathbf{V}\Delta_{\beta}\tilde{\mathbf{x}}$ *es correcta, es decir,* $\mathbf{V}\Delta_{\beta}\tilde{\mathbf{x}} = \mathbf{x}^{\omega}$

Demostración. Por hipótesis se tiene que $\mathbf{y}^{\mu} = \mathbf{x}^{\mu}\ \forall \mu \in \{1, 2, ..., p\}$ y, por consiguiente, $m = n$. Al establecer estas dos condiciones en el Teorema 4.13 (Ver referencia), se obtiene el resultado: $\mathbf{V}\Delta_{\beta}\tilde{\mathbf{x}} = \mathbf{x}^{\omega}$

El Teorema 4.30 nos dice que las memorias autoasociativas Alfa-Beta tipo V son inmunes a cierta cantidad de ruido aditivo.

Algoritmo Memorias Autoasociativas Alfa-Beta tipo Λ

Fase de Aprendizaje

Paso 1. Para cada $\mu = 1, 2, ..., p$, a partir de la pareja $(\mathbf{x}^{\mu},\mathbf{x}^{\mu})$ se construye la matriz

$$[\mathbf{x}^{\mu} \boxtimes (\mathbf{x}^{\mu})^t]_{mxn} \qquad \textbf{1.35}$$

Paso 2. Se aplica el operador binario máximo Λ a las matrices obtenidas en el paso 1:

$$\mathbf{\Lambda} = \bigvee_{\mu=1}^{p} [\mathbf{x}^{\mu} \boxtimes (\mathbf{x}^{\mu})^{t}] \qquad \textbf{1.36}$$

La entrada *ij*-ésima de la memoria está dada así:

$$\lambda_{ij} = \bigwedge_{\mu=1}^{p} \alpha\left(x_i^{\mu}, x_j^{\mu}\right) \qquad \textbf{1.37}$$

y de acuerdo con que α: A x $A \rightarrow B$, se tiene que $\lambda_{ij} \in B$, $\forall i \in \{1, 2, ..., n\}$. $\forall j \in \{1, 2, ..., n\}$.

Fase de Recuperación

La fase de recuperación de las memorias autoasociativas $\alpha\beta$ tipo Λ tiene dos casos posibles. En el primer caso el patrón de entrada es un patrón fundamental; es decir, la entrada es un patrón $\mathbf{x}^{\omega}$, con $\omega \in \{1, 2, ..., p\}$. En el segundo caso, el patrón de entrada NO es un patrón fundamental, sino la versión distorsionada de por lo menos uno de los patrones fundamentales; lo anterior significa que si el patrón de entrada es $\tilde{\mathbf{x}}$, debe existir al menos un valor de índice $\omega \in \{1, 2, ..., p\}$, que corresponde al patrón fundamental respecto del cual $\tilde{\mathbf{x}}$ es una versión alterada de alguno de los tres tipos: aditivo, sustractivo o mezclado.

CASO 1: Patrón fundamental. Se presenta a un patrón $\mathbf{x}^{\omega}$, con $\omega \in \{1, 2, ..., p\}$ a la memoria autoasociativa *Alfa-Beta* tipo **V** y se realiza la operación ∇_{β}:

$$\mathbf{\Lambda}\, \Delta_{\beta}\, \mathbf{x}^{\omega} \qquad \textbf{1.38}$$

El resultado de la operación anterior será el vector columna de dimensión *n*.

$$\left(\mathbf{\Lambda} \nabla_{\beta} \mathbf{x}^{\omega}\right)_i = \bigvee_{j=1}^{n} \beta(\lambda_{ij}, x_j^{\omega}) \qquad \textbf{1.39}$$

$$\left(\Lambda \nabla_{\beta} \mathbf{x}^{\omega}\right)_i = \bigvee_{j=1}^{n} \beta\left\{\left[\bigwedge_{\mu=1}^{p} \alpha(x_i^{\mu}, x_j^{\mu})\right], x_j^{\omega}\right\} \qquad \textbf{1.40}$$

CASO 2: Patrón alterado. Se presenta el patrón binario $\tilde{\mathbf{x}}$ (patrón alterado de algún patrón fundamental $\mathbf{x}^{\omega}$) que es un vector columna de dimensión *n*, a la memoria autoasociativa $\alpha\beta$ tipo $\mathbf{\Lambda}$ y se realiza la operación:

$$\Lambda \nabla_{\beta} \tilde{\mathbf{x}} \qquad \textbf{1.41}$$

Al igual que en el caso 1, el resultado de la operación anterior es un vector columna de dimensión *n*, cuya *i*-ésima componente se expresa de la siguiente manera:

$$\left(\Lambda \nabla_{\beta} \tilde{\mathbf{x}}\right)_i = \bigvee_{j=1}^{n} \beta(\lambda_{ij}, \tilde{x}_j) \qquad \textbf{1.42}$$

$$\left(\Lambda\nabla_{\beta}\tilde{\mathbf{x}}\right)_i = \bigvee_{j=1}^{n} \beta\left\{\left[\bigwedge_{\mu=1}^{p} \alpha(x_i^{\mu}, x_j^{\mu})\right], \tilde{x}_j\right\} \quad \textbf{1.43}$$

Lema 4.31. (*Numeración tal como aparece en* [20]). *Una memoria autoasociativa Alfa-Beta tipo Λ tiene únicamente unos en la diagonal principal.*

Demostración. La *ij*-ésima entrada de una memoria autoasociativa Alfa-Beta tipo $\boldsymbol{\Lambda}$ está dada por $\lambda_{ij} = \bigwedge_{\mu=1}^{p} \alpha\left(x_i^{\mu}, x_j^{\mu}\right)$. Las entradas de la diagonal principal se obtienen de la expresión anterior haciendo $i = j$:

$$\lambda_{ii} = \bigwedge_{\mu=1}^{p} \alpha\left(x_i^{\mu}, x_i^{\mu}\right), \quad \forall i \in \{1,2,...n\} \quad \textbf{1.44}$$

Por la propiedad de la tabla se tiene que $\alpha(x_i^{\mu}, x_i^{\mu}) = 1$, por lo que la expresión anterior se transforma en:

$$\lambda_{ii} = \bigwedge_{\mu=1}^{p} (1) = 1, \quad \forall i \in \{1,2,...n\} \quad \textbf{1.45}$$

Teorema 4.32. (*Numeración tal como aparece en* [20]). *Una memoria autoasociativa Alfa-Beta tipo $\boldsymbol{\Lambda}$ recupera de manera correcta el conjunto fundamental completo; además, tiene máxima capacidad de aprendizaje.*

Demostración. Sea $\omega = \{1, 2, ..., p\}$ arbitrario. De acuerdo con el lema 4.31, para cada $i \in \{1, ..., n\}$ escogida arbitrariamente $\exists\, j_0 = i \in \{1, ..., n\}$ que cumple con:

$$\lambda_{ii} = 1 = \alpha(x_i^{\omega}, x_i^{\omega}) \quad \textbf{1.46}$$

Es decir, para $i \in \{1, ..., n\}$ escogida arbitrariamente, $\exists\, j_0 = i \in \{1, ..., n\}$ que cumple con:

$$\lambda_{ij_0} = \alpha(x_i^{\omega}, x_{j_0}^{\omega}) \quad \textbf{1.47}$$

Por lo tanto, de acuerdo con el Teorema 4.20

$$\boldsymbol{\Lambda\nabla}_{\beta}\, \mathbf{x}^{\omega} = \mathbf{x}^{\omega}, \forall \omega \in \{1, 2, ..., p\} \quad \textbf{1.48}$$

Esto significa que la memoria autoasociativa Alfa-Beta tipo Λ recupera de manera correcta el conjunto fundamental completo.

Además, en la demostración de este Teorema, en ningún momento aparece restricción alguna sobre *p* que es la cardinalidad del conjunto fundamental; y esto quiere decir que el conjunto fundamental puede crecer tanto como se quiera. La consecuencia directa es que el número de patrones que puede aprender una memoria autoasociativa Alfa-Beta tipo Λ, con recuperación correcta, es máximo.

El Teorema 4.32 se puede enunciar desde un enfoque matricial de la siguiente manera: dado que para cada asociación ($\mathbf{x}^{\omega}$,$\mathbf{x}^{\omega}$) del conjunto fundamental de una memoria

autoasociativa Alfa-Beta $\boldsymbol{\Lambda}$, se cumple que cada fila de la matriz $\mathbf{x}^{\omega} \boxtimes (\mathbf{x}^{\omega})^{t} - \boldsymbol{\Lambda}$ contiene una entrada cero, entonces de la memoria $\boldsymbol{\Lambda}$ recupera el conjunto completo de patrones fundamentales en forma correcta.

Teorema 4.33. (*Numeración tal como aparece en* [20]). *Sea* $\{(\mathbf{x}^{\mu},\mathbf{x}^{\mu}) \mid \mu = 1, 2, ..., p\}$ *el conjunto fundamental de una memoria autoasociativa Alfa-Beta representada por* $\boldsymbol{\Lambda}$, *y sea* $\tilde{\mathbf{x}} \in A^{n}$ *un patrón alterado con ruido sustractivo respecto a algún patrón fundamental* $\mathbf{x}^{\omega}$ con $\omega \in \{1, 2, ..., p\}$. *Si se presenta* $\tilde{\mathbf{x}}$ *a la memoria* $\boldsymbol{\Lambda}$ *como entrada, y si además para cada* $i \in \{1, ..., n\}$ *se cumple la condición de que* $\exists j = j_0 \in \{1, ..., n\}$, *el cual depende de* ω *y de i tal que* $\lambda_{ij_0} \leq \alpha(x^{\omega}, \tilde{x}_{j_0})$, *entonces la recuperación* $\Lambda\nabla_{\beta}\tilde{\mathbf{x}}$ *es correcta, es decir,* $\Lambda\nabla_{\beta}\tilde{\mathbf{x}} = \mathbf{x}^{\omega}$

Demostración. Por hipótesis se tiene que $\mathbf{y}^{\mu} = \mathbf{x}^{\mu}$ $\forall \mu \in \{1, 2, ..., p\}$ y, por consiguiente, $m = n$. Al establecer estas dos condiciones en el Teorema 4.23 (Ver referencia), se obtiene el resultado:

$$\Lambda\nabla_{\beta}\tilde{\mathbf{x}} = \mathbf{x}^{\omega} \qquad \textbf{1.49}$$

El Teorema 4.33 nos dice que las memorias autoasociativas Alfa-Beta tipo V son inmunes a cierta cantidad de ruido sustractivo.

CAPÍTULO 4. La propuesta

En este capítulo, que consta de tres secciones, se presenta la propuesta central de esta tesis. La sección 4.1 contiene la modificación que se realizó al algoritmo original de las memorias heteroasociativas Alfa-Beta *Max y Min*, tanto en la fase de aprendizaje como en la fase de recuperación, que tiene como principal ventaja recuperar el conjunto fundamental completo; se incluyen definiciones, lemas y teoremas que sustentan las modificaciones hechas al modelo original, además de ejemplos representativos.

En la sección 4.2 se propone el modelo de multimemoria heteroasociativa Alfa-Beta, que servirá de base para la siguiente sección, la 4.3, donde se propone y ejemplifica un algoritmo de clasificación de patrones que utiliza como base los algoritmos propuestos en los primeras dos secciones; dicho algoritmo, de acuerdo con el objetivo central de este trabajo de tesis, será aplicado en problemas de clasificación en Bioinformática.

1.1 Nuevo Algoritmo de Memoria Heteroasociativa Alfa-Beta

En la teoría de las memorias asociativas Alfa-Beta se garantiza, mediante los teoremas 4.28 y 4.32 de la referencia [20], la recuperación completa del conjunto fundamental sólo en el caso de las memorias autoasociativas. Para el caso de las memorias heteroasociativas, dicho comportamiento no se puede garantizar; es por ello que en esta sección se propone un algoritmo que modifique el original, de modo que se recupere el conjunto fundamental completo.

Por principio es necesario redefinir las fases de aprendizaje y recuperación de las memorias heteroasociativas *Max* y *Min*, creando así el nuevo algoritmo. Para ello comenzamos con una serie de definiciones que nos serán útiles para la demostración de los lemas y teoremas tanto para las Memorias Heteroasociativas tipo *Max* como para las *Min*.

Definición 1: Sean $h, n \in Z^+, A = \{0,1\}$ y sea $\mathbf{x}^h \in A^n$. Se denota a la suma de los componentes positivos de $\mathbf{x}^h$ de la siguiente manera:

$$U_h = \sum_{j=1}^{n} x_j^h \qquad \textbf{1.1}$$

Ejemplo 1: Calcular U_h para cada uno de los patrones de entrada x^1, x^2, x^3 y x^4

$$x^1 = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}, x^2 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, x^3 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, x^4 = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 1 \end{pmatrix}$$

de acuerdo con la definición 1 la suma de los componentes positivos correspondiente a cada uno de los vectores, es la siguiente:

$$U_1 = 3, U_2 = 1, U_3 = 2, U_4 = 4$$

Definición 2: Sea **V** una memoria heteroasociativa Alfa-Beta tipo Max y su correspondiente conjunto fundamental expresado por $\{(\mathbf{x}^\mu, \mathbf{y}^\mu) | \mu = 1, 2, ...p\}$ $\mathbf{x}^\mu \in A^n$ y $\mathbf{y}^\mu \in A^p$, con $A = \{0,1\}, B = \{0,1,2\}$, $n \in Z^+$. La suma de las componentes iguales a 1 de la $i-ésima$ fila de la matriz **V** está dada por:

$$s_i = \sum_{j=1}^{n} T_j \qquad \textbf{1.2}$$

donde $T \in B^n$ y las componentes de T se definen así:

$$T_i = \begin{cases} 1 & \leftrightarrow \quad v_{ij} = 1 \\ 0 & \leftrightarrow \quad v_{ij} \neq 1 \end{cases} \qquad \textbf{1.3}$$

$\forall j \in \{1, 2, ..., n\}$. Las p componentes s_i conforman el *vector suma max* $\mathrm{s} \in \mathrm{Z}^{\mathrm{p}}$.

Ejemplo 2: Si **V** es una memoria heteroasociativa Alfa-Beta Tipo Max como se muestra a continuación:

$$\mathbf{V} = \begin{pmatrix} 1 & 0 & 1 & 1 & 2 \\ 0 & 1 & 2 & 2 & 0 \\ 1 & 2 & 0 & 1 & 2 \\ 1 & 1 & 0 & 1 & 1 \end{pmatrix}$$

el *vector suma max* queda expresado de la siguiente manera:

$$\mathbf{s} = \begin{pmatrix} 3 \\ 1 \\ 2 \\ 4 \end{pmatrix}$$

Definición 3: Sean $\alpha, \beta, n \in Z^+, A = \{0,1\}$ y $\mathbf{x}^\alpha, \mathbf{x}^\beta \in A^n$ dos vectores; entonces $\mathbf{x}^\alpha \leq \mathbf{x}^\beta \leftrightarrow x_i^\alpha \leq x_i^\beta \ \forall i \in \{1,2,\ldots,n\}$ y además $\mathbf{x}^\alpha < \mathbf{x}^\beta \leftrightarrow \forall i x_i^\alpha \leq x_i^\beta$ y $\exists j$ tal que $x_j^\alpha < x_j^\beta$.

Ejemplo 3: Tomemos los vectores x^3 y x^4 del ejemplo 1. De acuerdo con la definición 3 $x^3 < x^4$ ya que

$$\begin{array}{ccc} 1 & \leq & 1 \\ 0 & < & 1 \\ 0 & \leq & 0 \\ 1 & \leq & 1 \\ 0 & < & 1 \end{array}$$

por otro lado, si relacionamos el vector x^4 consigo mismo, se cumple que $x^4 \leq x^4$

$$\begin{array}{ccc} 1 & \leq & 1 \\ 1 & \leq & 1 \\ 0 & \leq & 0 \\ 1 & \leq & 1 \\ 1 & \leq & 1 \end{array}$$

Definición 4: Sean $\alpha, n \in Z^+, A = \{0,1\}$ y sea $\mathbf{x}^\alpha \in A^n$; entonces el vector negado de $\mathbf{x}^\alpha$ se denota por $\sim \mathbf{x}^\alpha$ y sus componentes se definen así:

$$\sim x_i^\alpha = \begin{cases} 1 & x_i^\alpha = 0 \\ 0 & x_i^\alpha = 1 \end{cases} \qquad \textbf{1.4}$$

$\forall i \in \{1,2,\ldots,n\}.$

Ejemplo 4: Sea el patrón x^1 del ejemplo 1; de acuerdo con la definición 4 su correspondiente patrón negado $\sim x^1$ es el siguiente:

$$\sim x^1 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

Definición 5: Sean $\alpha, \beta, n \in Z^+, A = \{0,1\}$ y sea $\mathbf{x}^h \in A^n$. Denotamos a la suma de los componentes iguales a 0 de $\mathbf{x}^h$ como:

$$C_h = \sum_{j=1}^{n} \sim x_j^h \qquad \textbf{1.5}$$

Ejemplo 5: Calcular C_h para los patrones de x^1, x^2, x^3 y x^4 del ejemplo 1:

$$x^1 = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}, x^2 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, x^3 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, x^4 = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 1 \end{pmatrix} \qquad C_1 = 2, C_2 = 4, C_3 = 3, C_4 = 1$$

Definición 6: Sea Λ una memoria heteroasociativa Alfa-Beta tipo *Min* y su correspondiente conjunto fundamental expresado por $\{(\mathbf{x}^\mu, \mathbf{y}^\mu) | \mu = 1, 2, ...p\}$, $\mathbf{x}^\mu \in A^n$ y $\mathbf{y}^\mu \in A^p$, con $A = \{0,1\}$, $n \in Z^+$. La suma de las componentes iguales a 0 de la $i-ésima$ fila de la matriz Λ está dada por:

$$r_i = \sum_{j=1}^{n} \sim T_j \qquad \textbf{1.6}$$

donde $T \in B^n$ y las componentes de T se definen así:

$$T_i = \begin{cases} 1 & \leftrightarrow \quad \lambda_{ij} = 0 \\ 0 & \leftrightarrow \quad \lambda_{ij} \neq 0 \end{cases} \qquad \textbf{1.7}$$

$\forall j \in \{1, 2, ..., n\}$. Las p componentes r_i conforman el *vector suma min* $\mathrm{r} \in \mathrm{Z}^{\mathrm{p}}$.

Ejemplo 6: Si Λ es una memoria heteroasociativa Alfa-Beta Tipo Min como se muestra a continuación:

$$\Lambda = \begin{pmatrix} 0 & 2 & 0 & 1 & 0 \\ 1 & 0 & 1 & 2 & 1 \\ 0 & 1 & 1 & 0 & 2 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

el *vector suma min* queda expresado de la siguiente manera:

$$r = \begin{pmatrix} 3 \\ 1 \\ 2 \\ 4 \end{pmatrix}$$

1.1.1 Nueva Memoria Heteroasociativa Alfa-Beta Tipo *Max*

Fase de Aprendizaje

Sean $A=\{0,1\}, n, p \in Z^+, \mu \in \{1,2,...p\}, i \in \{1,2,...,p\}$ y $j \in \{1,2,...n\}$, y sean $\mathbf{x} \in A^n$ y $\mathbf{y} \in A^p$ los vectores de entrada y salida, respectivamente. A partir de estos vectores es posible construir el conjunto fundamental expresado como $\{(\mathbf{x}^\mu, \mathbf{y}^\mu) \mid \mu = 1,2,..., p\}$.

El nuevo modelo requiere que el vector $\mathbf{y}$ sea creado de acuerdo con la codificación *one-hot*; es decir, $y_k^\mu = 1$ y $y_j^\mu = 0$ para $j = 1,2,\ldots,k-1,k+1,\ldots,p$ tal que $k \in \{1,2,\ldots,p\}$. Además, es necesario que a cada vector $\mathbf{y}^\mu$ le corresponda *uno y sólo uno* de los vectores $\mathbf{x}^\mu$; es decir, que en el conjunto fundamental exista *una y sólo una* pareja $(\mathbf{x}^\mu, \mathbf{y}^\mu)$.

PASO 1

Para cada $\mu \in \{1,2,...p\}$, a partir de la pareja ordenada $(\mathbf{x}^\mu, \mathbf{y}^\mu)$ se construye la matriz

$$[\mathbf{y}^\mu \boxtimes (\mathbf{x}^\mu)^t]_{mxn} \qquad \textbf{1.8}$$

PASO 2

Se aplica el operador binario máximo $\vee$ a las matrices obtenidas en el paso 1, para obtener la memoria heteroasociativa Alfa-Beta tipo *Max*

$$\mathbf{V} = \bigvee_{\mu=1}^{p} [\mathbf{y}^\mu \boxtimes (\mathbf{x}^\mu)^t] \qquad \textbf{1.9}$$

de tal modo que la $ij-ésima$ entrada esté dada por

$$v_{ij} = \bigvee_{\mu=1}^{p} \alpha(y_i^\mu, x_j^\mu) \qquad \textbf{1.10}$$

Fase de Recuperación

PASO 1

Al presentarse un patrón $\mathbf{x}^\varpi$, con $\varpi \in \{1,2,...p\}$ a $\mathbf{V}$ se realiza la operación Δ_β, cuyo resultado será asignado a un vector que llamaremos $\mathbf{z}^\varpi$:

$$\mathbf{z}^\varpi = \mathbf{V} \Delta_\beta \mathbf{x}^\varpi \qquad \textbf{1.11}$$

Por lo que la $i-ésima$ componente del vector columna resultante se expresa como:

$$\mathbf{z}_i^{\varpi} = \bigwedge_{j=1}^{n} \beta(v_{ij}, x_j^{\varpi}) \qquad \textbf{1.12}$$

PASO 2

Una vez que se tiene el vector columna $\mathbf{z}^{\varpi}$, es necesario construir el *vector suma max* $\mathbf{s}$, el cual contiene en su $i-ésima$ componente la suma de las componentes de la $i-ésima$ fila de la matriz $\mathbf{V}$:

$$s_i = \sum_{j=1}^{n} T_j \qquad \textbf{1.13}$$

donde $T \in B^n$ y las componentes de T se definen así:

$$T_i = \begin{cases} 1 & \leftrightarrow \quad v_{ij} = 1 \\ 0 & \leftrightarrow \quad v_{ij} \neq 1 \end{cases} \qquad \textbf{1.14}$$

$\forall j \in \{1, 2, ..., n\}.$ por lo tanto la correspondiente $\mathbf{y}^{\varpi}$ está dada por:

$$\mathbf{y}_i^{\varpi} = \begin{cases} 1 & \text{si} \quad s_i = \bigvee_{k \in \theta} s_k \quad \wedge \quad z_i^{\varpi} = 1 \\ 0 & \text{en otro caso} \end{cases} \qquad \textbf{1.15}$$

donde $\theta = \{i \mid Z_i^{\varpi} = 1\}$.

A continuación se presentan los lemas y teoremas que soportan el nuevo algoritmo.

Lema 1: Sea $\mathbf{x}^i \in A^n$ un patrón tomado arbitrariamente del conjunto fundamental. En la fase de aprendizaje de la memoria heteroasociativa tipo Max, $\mathbf{x}^i$ contribuye sólo en la $i-ésima$ fila de la matriz $\mathbf{V}$ con U_i veces el valor 1 y $(n - U_i)$ veces el valor 2.

Demostración: Sea $\mathbf{x}^k \in A^n$ y $\mathbf{y}^k \in A^p$ con $A = \{0,1\}, k, n, p \in Z^+$ dos patrones fundamentales, tomados arbitrariamente, que forman la $k-ésima$ asociación $(\mathbf{x}^k, \mathbf{y}^k)$ de la memoria $\mathbf{V}$. De acuerdo con la fase de aprendizaje sabemos que la matriz $\mathbf{V}$ está dada por

$$\mathbf{V} = \bigvee_{\mu=1}^{p} [\mathbf{y}^{\mu} \boxtimes (\mathbf{x}^{\mu})^t] \qquad \textbf{1.16}$$

de manera particular para la $k-ésima$ asociación

$$[\mathbf{y}^k \boxtimes (\mathbf{x}^k)^t]_{ij} = \alpha(\mathbf{y}_i^k, \mathbf{x}_j^k) \qquad \textbf{1.17}$$

Ahora, dada la manera en que $\mathbf{y}^k$ es construida, sucede que

$$\begin{aligned}&\forall i \in \{1,2,...,k-1,k+1,...p\}, \forall j \in \{1,2,...n\}, k \in \{1,2,...p\} \\ &\mathbf{y}_k^k = 1 \rightarrow \alpha(\mathbf{y}_k^k, \mathbf{x}_j^k) = 1 \vee \alpha(\mathbf{y}_k^k, \mathbf{x}_j^k) = 2 \\ &\mathbf{y}_i^k = 0 \rightarrow \alpha(\mathbf{y}_i^k, \mathbf{x}_j^k) = 1 \vee \alpha(\mathbf{y}_i^k, \mathbf{x}_j^k) = 0\end{aligned} \quad \textbf{1.18}$$

de acuerdo con la expresión 4.8 es evidente que los máximos valores de las componentes de la $k-ésima$ matriz están sólo en la $k-ésima$ fila, incluyen sólo los valores 1 o 2, y dependen exclusivamente de los valores en las componentes de $\mathbf{x}^k$, esto es, cuando $\mathbf{x}_j^k = 1 \leftrightarrow \alpha(\mathbf{y}_k^k, \mathbf{x}_j^k) = 1$ o $\mathbf{x}_j^k = 0 \leftrightarrow \alpha(\mathbf{y}_k^k, \mathbf{x}_j^k) = 2.$ Por lo tanto, dado que las asociaciones del conjunto fundamental son creadas de tal manera que a *un patrón de entrada le corresponde uno y sólo uno de los patrones de salida* y que $k \in \{1,2,...p\}$ fue tomada de manera arbitraria se puede afirmar que la matriz $\mathbf{V}$ es afectada en su $i-ésima$ fila por el patrón $\mathbf{x}^i$ y lo hace con U_i valores 1 y $(n-U_i)$ valores 2. Por último podemos reescribir la fase de aprendizaje con la siguiente expresión:

$$\begin{aligned}&\forall i \in \{1,2,..., p\} \quad \text{y} \quad \forall j \in \{1,2,..., n\} \\ &v_{ij} = \alpha(\mathbf{y}_i^i, \mathbf{x}_j^i)\end{aligned} \quad \textbf{1.19}$$

Lema 2: Sea s el vector suma de la memoria heteroasociativa $\mathbf{V}$; entonces, $s_i = U_i$ $\forall i \in \{1,2,...p\}$.

Demostración*:* Sea s el vector suma de la memoria heteroasociativa $\mathbf{V}$, cuya i-ésima componente está expresada como sigue:

$$s_i = \sum_{j=1}^{n} T_j \quad \textbf{1.20}$$

donde $T \in B^n$ y las componentes de T se definen así:

$$T_i = \begin{cases} 1 & \leftrightarrow \quad v_{ij} = 1 \\ 0 & \leftrightarrow \quad v_{ij} \neq 1 \end{cases} \quad \textbf{1.21}$$

$\forall j \in \{1,2,...,n\}.$

Por otro lado, sabemos por la definición 1 que:

$$U_h = \sum_{j=1}^{n} x_j^h \quad \textbf{1.22}$$

dado que $\mathbf{x}^i \in A^n, A \in \{0,1\}$ la expresión 4.22 se puede reescribir como sigue:

$$U_i = \sum_{j=1}^{n} x_j^i \quad \textbf{1.23}$$

Además, sabemos por el lema 1 en su expresión 4.19 que $\mathbf{x}^i$ afecta a la matriz $\mathbf{V}$ sólo en su $i-ésima$ fila, por lo que es posible expresar a 4.20 de la siguiente manera:

$$s_i = \sum_{j=1}^{n} \alpha(y_{i,}^{i} x_j^i) \qquad \textbf{1.24}$$

y dada la forma en que fue construido el vector $\mathbf{y}$, $y_i^i = 1 \forall i \in \{1,2,...,p\}$, por lo que la expresión $\alpha(y_{i,}^{i} x_j^i)$ depende únicamente de x_j^i y, de acuerdo con el lema 1:

$$s_i = \sum_{j=1}^{n} x_j^i \qquad \textbf{1.25}$$

Por último por transitividad de las expresiones 4.23 y 4.25 concluimos que

$$s_i = \left(\sum_{j=1}^{n} x_j^i \right) = Ui \qquad \textbf{1.26}$$

Lema 3: Sea $\mathbf{V}$ una memoria heteroasociativa tipo *Max* cuyo conjunto fundamental está expresado como sigue $\{(\mathbf{x}^\mu, \mathbf{y}^\mu) | \mu = 1,2,...p\}$ con $\mathbf{x}^\varpi \in A^n, A \in \{0,1\}, \varpi \in \{1,2,...,p\}$ y $n, p \in Z^+$ el patrón fundamental que será presentado a la memoria $\mathbf{V}$. Después de la fase de recuperación original del algoritmo de memoria heteroasociativa Alfa-Beta tipo *Max*, se obtendrá un vector columna $\mathbf{z}^\varpi \in A^p$ el cual presenta el valor 1 en cada componente cuyo índice $i \in Z^+$ es el índice de la fila de la memoria $\mathbf{V}$ que corresponde a los patrones fundamentales menores o iguales a $\mathbf{x}^\varpi$; dicho de otra forma:

$$\forall i \, z_i^\varpi = 1 \rightarrow x^i \leq x^\omega, x^i \in \{(\mathbf{x}^\mu, \mathbf{y}^\mu) | \mu = 1,2,...p\} \qquad \textbf{1.27}$$

Demostración*:* Dada la fase de recuperación original de las memorias heteroasociativas tipo *Max*, sabemos que

$$z_i^\varpi = 1 \rightarrow \bigwedge_{j=1}^{n} \beta(v_{ij}, x_j^\varpi) = 1 \qquad \textbf{1.28}$$

y para que $\bigwedge_{j=1}^{n} \beta(v_{ij}, x_j^\varpi) = 1$, como la función β sólo puede arrojar 0 o 1, es necesario que $\forall j \in \{1,2,...,n\}$ $\beta(v_{ij}, x_j^\varpi) = 1$. Ahora, a partir de lo anterior y por lema 1, sólo se pueden dar los siguientes casos para $\forall j \in \{1,2,...,n\}$.

$$\beta(v_{ij}, x_j^\varpi) = 1 \rightarrow \begin{cases} v_{ij} = 1 \wedge x_j^\varpi = 1 \\ v_{ij} = 2 \wedge (x_j^\varpi = 1 \vee x_j^\varpi = 0) \end{cases} \qquad \textbf{1.29}$$

Ahora, dado que sabemos por el lema 1 que cada patrón x^i afecta sólo a la $i-ésima$ fila de la memoria $\mathbf{V}$ y lo hace conforme a la fase de aprendizaje, de la expresión 4.29 podemos inferir lo siguiente:

$\forall j$ siempre que suceda $(v_{ij} = 1 \wedge x_j^{\varpi} = 1)$ o $(v_{ij} = 2 \wedge x_j^{\varpi} = 0)$ podemos concluir que $x^i = x^{\varpi}$; además, $\forall j$ siempre que suceda $(v_{ij} = 1 \wedge x_j^{\varpi} = 1)$ o $\exists j(v_{ij} = 2 \wedge x_j^{\varpi} = 1)$ podemos concluir que $x^i < x^{\varpi}$. Es decir

$$x^i \leq x^{\varpi} \qquad \textbf{1.30}$$

por lo tanto, por transitividad de 4.28, 4.29 y 4.30 podemos concluir

$$\forall i \, z_i^{\varpi} = 1 \rightarrow x^i \leq x^{\omega}, x^i \in \left\{ \left(\mathbf{x}^{\mu}, \mathbf{y}^{\mu} \right) | \, \mu = 1, 2, ... p \right\} \qquad \textbf{1.31}$$

Teorema 1: Sea $\mathbf{V}$ una memoria heteroasociativa tipo *Max* con su conjunto fundamental expresado por $\left\{ \left(\mathbf{x}^{\mu}, \mathbf{y}^{\mu} \right) | \, \mu = 1, 2, ... p \right\}$, el cual no contiene asociaciones repetidas. Sea $\mathbf{x}^{\varpi} \in A^n, A \in \{0,1\}, \varpi \in \{1,2,..., p\}$ un patrón fundamental, el cual a sido presentado como patrón de entrada a la matriz $\mathbf{V}$, y $\mathbf{z}^{\varpi} \in A^p$ el vector resultante de la fase original de recuperación de las memorias heteroasociativas tipo *Max* con $A \in \{0,1\}, n, p \in Z^+$. La fase de recuperación del nuevo algoritmo de memoria heteroasiciativa tipo *Max* siempre presentará recuperación correcta; esto es, con el algoritmo propuesto siempre obtendremos la correspondiente $\mathbf{y}^{\varpi}$ sin ambigüedad.

Demostración: El fase de recuperación del nuevo algoritmo de memoria heteroasociativa tipo *Max* está expresada de la siguiente manera

$$\mathbf{y}_i^{\varpi} = \begin{cases} 1 & \text{si} \quad s_i = \bigvee_{k \in \theta} s_k \quad \wedge \quad z_i^{\varpi} = 1 \\ 0 & \textit{en otro caso} \end{cases} \qquad \textbf{1.32}$$

donde $\theta = \left\{ i \mid Z_i^{\varpi} = 1 \right\}$.

Para demostrar la recuperación correcta del nuevo algoritmo es necesario asegurarnos de que, para todas las componentes en donde $Z_i^{\varpi} = 1$, sólo existe un valor máximo en las componentes de s_i y corresponde al patrón correcto. Esto puede ser demostrado por *contradicción.*

Supongamos que $x^{\alpha} \in A^n$ y $x^{\beta} \in A^n$ son los patrones correspondientes a $z_{\alpha}^{\varpi} = 1$ y $z_{\beta}^{\varpi} = 1$ al presentarle un patrón $x^{\varpi} \in A^n$ a la memoria $\mathbf{V}$ con $x^{\alpha}, x^{\beta}, x^{\varpi} \in \left\{ \left(\mathbf{x}^{\mu}, \mathbf{y}^{\mu} \right) | \, \mu = 1, 2, ... p \right\}$ y siendo x^{α} el patrón correcto y x^{β} un patrón espurio arbitrario. Además, se tiene que para cada patrón sus correspondientes valores de s_i, particularmente s_{α} y s_{β}, mantiene que $s_{\alpha} > s_{\beta}$. Ahora asumimos la negación de lo que queremos probar

$$s_{\alpha} \leq s_{\beta} \qquad \textbf{1.33}$$

De acuerdo al lema 2 sabemos que la expresión 4.33 puede ser expresada de la siguiente manera:

$$U_\alpha \leq U_\beta \quad \textbf{1.34}$$

por el lema 3 sabemos que cada patrón espurio x^i, donde x^ϖ es el correcto implica $\forall iz_i^\varpi = 1, \mathbf{x}^i \neq \mathbf{x}^\varpi \rightarrow \mathbf{x}^i < \mathbf{x}^\varpi$ por lo tanto podemos tomar por hipótesis que:

$$\mathbf{x}^\beta < \mathbf{x}^\alpha \quad \textbf{1.35}$$

Y, por lo tanto, de acuerdo a la definición 1 podemos expresar a 4.35 como

$$U_\beta < U_\alpha \quad \textbf{1.36}$$

lo cual nos lleva a una contradicción entre 4.34 y 4.36, por lo mismo podemos afirmar que $U_\alpha \leq U_\beta$ no puede ser cierto por lo que $U_\alpha > U_\beta$ es verdadera. Y dado que $\mathbf{x}^\beta$ fue tomada de manera arbitraria, esta condición es cierta para todo patrón espurio.

El siguiente ejemplo nos muestra la fase de aprendizaje y recuperación del nuevo algoritmo de memoria heteroasociativa Alfa-Beta.

Ejemplo 7: Sean los patrones de entrada

$$x^1 = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}, x^2 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, x^3 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, x^4 = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 1 \end{pmatrix}$$

y sus correspondientes clases

$$y^1 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, y^2 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, y^3 = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, y^4 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

Las clases están construidas de acuerdo a la codificación one-hot y a cada clase le corresponde uno y sólo uno de los patrones de entrada, por lo que el conjunto fundamental queda de la siguiente manera:

$$\{(x^1, y^1), (x^2, y^2), (x^3, y^3), (x^4, y^4)\}$$

Paso 1:

$$\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \boxtimes \begin{pmatrix} 1 & 0 & 1 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 1 & 1 & 2 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \boxtimes \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 & 1 & 1 \\ 2 & 1 & 2 & 2 & 2 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \boxtimes \begin{pmatrix} 1 & 0 & 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 2 & 2 & 1 & 2 \\ 0 & 1 & 1 & 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \boxtimes \begin{pmatrix} 1 & 1 & 0 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 2 & 1 & 1 \end{pmatrix}$$

Paso 2: Se aplica el operador binario máximo $\vee$ a las matrices obtenidas en el paso anterior.

$$\mathbf{V} = \begin{pmatrix} 1 & 2 & 1 & 1 & 2 \\ 2 & 1 & 2 & 2 & 2 \\ 1 & 2 & 2 & 1 & 2 \\ 1 & 1 & 2 & 1 & 1 \end{pmatrix}$$

Fase de Recuperación:

Al presentarle un patrón $\mathbf{x}^{\varpi}$ con $\varpi \in \{1, 2, ..., p\}$, particularmente x^4, a la memoria $\mathbf{V}$ obtenida en la fase de aprendizaje, de acuerdo con el paso 1 del algoritmo de memorias heteroasociativas tipo *Max* propuesto en esta tesis, tenemos que:

$$\mathbf{V}\Delta_\beta x^4 = \begin{pmatrix} 1 & 2 & 1 & 1 & 2 \\ 2 & 1 & 2 & 2 & 2 \\ 1 & 2 & 2 & 1 & 2 \\ 1 & 1 & 2 & 1 & 1 \end{pmatrix} \Delta_\beta \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

De acuerdo con el paso 2 de la fase de recuperación de las memorias heteroasociativas tipo *Max* propuesto es este trabajo, se obtiene el vector resultante denominado z^4; y posteriormente se crea el vector $\mathbf{s}$.

$$z^4 = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 1 \end{pmatrix}, \mathbf{s} = \begin{pmatrix} 3 \\ 1 \\ 2 \\ 4 \end{pmatrix}$$

Ahora, según la expresión del paso 2 de la fase de recuperación del nuevo algoritmo de memorias heteroasociativas tipo *Max*:

$$y^4 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

ya que el máximo valor de $\mathbf{s}_j$ donde $z_j^4 = 1 \forall j \in \{1,2,3,4\}$ es 4.

1.1.2 Nueva Memoria Heteroasociativa Alfa-Beta Tipo *Min*

Fase de Aprendizaje

Sean $A = \{0,1\}$, $n, p \in Z^+$, $\mu \in \{1,2,...p\}, i \in \{1,2,...,p\}$ y $j \in \{1,2,...n\}$, y sea $\mathbf{x} \in A^n$ y $\mathbf{y} \in A^p$ vectores de entrada y de salida, respectivamente. A partir de estos vectores es posible construir el conjunto fundamental expresado por $\{(\mathbf{x}^\mu, \mathbf{y}^\mu) \mid \mu = 1,2,...,p\}$.

El nuevo modelo requiere que el vector $\mathbf{y}$ sea creado de acuerdo con la codificación *cero-hot*, es decir, $y_k^\mu = 0$, y $y_j^\mu = 1$ para $j = 1,2,\ldots,k-1,k+1,\ldots,p$ tal que $k \in \{1,2,\ldots,p\}$. Además, es necesario que para a cada vector $\mathbf{y}^\mu$ le corresponda *un y sólo un* vector $\mathbf{x}^\mu$; es decir, que en el conjunto fundamental exista *una y sólo una* pareja $(\mathbf{x}^\mu, \mathbf{y}^\mu)$.

PASO 1

Para cada $\mu \in \{1,2,...p\}$, a partir de la pareja ordenada $(\mathbf{x}^\mu, \mathbf{y}^\mu)$ se construye la matriz

$$[\mathbf{y}^{\mu} \boxtimes (\mathbf{x}^{\mu})^{t}]_{mxn} \quad \textbf{1.37}$$

PASO 2

Se aplica el operador binario mínimo $\wedge$ a las matrices obtenidas en el paso anterior, para obtener la memoria heteroasociativa Alfa-Beta tipo *Min*

$$\mathbf{\Lambda} = \bigwedge_{\mu=1}^{p} [\mathbf{y}^{\mu} \boxtimes (\mathbf{x}^{\mu})^{t}] \quad \textbf{1.38}$$

del tal modo que la entrada $ij-ésima$ esté dada por

$$\lambda_{ij} = \bigwedge_{\mu=1}^{p} \alpha(y_i^{\mu}, x_j^{\mu}) \quad \textbf{1.39}$$

Fase de Recuperación:

PASO 1

Al presentarle un patrón $\mathbf{x}^{\varpi}$ con $\varpi \in \{1,2,...p\}$ $\mathbf{\Lambda}$ a realiza la operación ∇_{β}, cuyo resultado será asignado a un vector que llamaremos $\mathbf{z}^{\varpi}$:

$$\mathbf{z}^{\varpi} = \mathbf{\Lambda} \nabla_{\beta} \mathbf{x}^{\varpi} \quad \textbf{1.40}$$

Por lo que la $i-ésima$ componente del vector columna resultante se expresa como:

$$\mathbf{z}_{\mathbf{i}}^{\varpi} = \bigvee_{j=1}^{n} \beta(\lambda_{ij}, x_j^{\varpi}) \quad \textbf{1.41}$$

PASO 2

Una vez que se tiene el vector columna $\mathbf{z}^{\varpi}$, es necesario construir el *vector suma min* $\mathbf{r} \in Z^{p}$ el cual contiene en su $i-ésima$ componente el número de ceros de la $i-ésima$ fila de la matriz $\mathbf{\Lambda}$.

$$r_i = \sum_{j=1}^{n} \sim T_j \quad \textbf{1.42}$$

donde $T \in B^{n}$ y las componentes de T se definen así:

$$T_i = \begin{cases} 0 & \leftrightarrow \quad \lambda_{ij} = 0 \\ 1 & \leftrightarrow \quad \lambda_{ij} \neq 0 \end{cases} \quad \textbf{1.43}$$

$\forall j \in \{1,2,...,n\}$.

por la tanto la correspondiente $\mathbf{y}^{\varpi}$ esta dada por:

$$\mathbf{y}_i^{\varpi} = \begin{cases} 0 & \text{si} \quad r_i = \bigwedge_{k\in\theta} r_k \quad \wedge \quad z_k^{\varpi} = 0 \\ 1 & \text{en otro caso} \end{cases} \qquad \textbf{1.44}$$

donde $\theta = \{i \mid Z_i^{\varpi} = 0\}$.

A continuación se presentan los lemas y teoremas que soportan el algoritmo propuesto.

Lema 4: Sea $\mathbf{x}^i \in A^n$ un patrón, tomado arbitrariamente del conjunto fundamental con $i, n \in Z^+$. En la fase de aprendizaje de la memoria heteroasociativa tipo Min, el vector $\mathbf{x}^i$ contribuye sólo en la $i-ésima$ fila de la matriz $\mathbf{\Lambda}$ y contribuye con U_i veces el valor 0 y $(n-U_i)$ veces el valor 1.

Demostración: Sea $\mathbf{x}^k \in A^n$ y $\mathbf{y}^k \in A^p$ con $k \in \{1,2,...,p\}$, $A = \{0,1\}, n \in Z^+$ y $p \in Z^+$ dos patrones fundamentales, tomados arbitrariamente, que forman la $k-ésima$ asociación $(\mathbf{x}^k, \mathbf{y}^k)$ de la memoria $\mathbf{\Lambda}$. De acuerdo con la fase de aprendizaje sabemos que la matriz $\mathbf{\Lambda}$ esta dada por:

$$\mathbf{\Lambda} = \bigwedge_{\mu=1}^{p} [\mathbf{y}^{\mu} \boxtimes (\mathbf{x}^{\mu})^t] \qquad \textbf{1.45}$$

de manera particular para la $k-ésima$ asociación

$$[\mathbf{y}^k \boxtimes (\mathbf{x}^k)^t]_{ij} = \alpha(\mathbf{y}_i^k, \mathbf{x}_j^k) \qquad \textbf{1.46}$$

Ahora, dada la manera en que $\mathbf{y}^k$ es construida y que $\mathbf{x}^k \in A$, sucede que

$$\forall i \in \{1,2,...,k-1,k+1,...p\}, \forall j \in \{1,2,...n\}, k \in \{1,2,...p\}$$

$$\begin{aligned} \mathbf{y}_k^k = 0 &\rightarrow \alpha(\mathbf{y}_k^k, \mathbf{x}_j^k) = 1 \vee \alpha(\mathbf{y}_k^k, \mathbf{x}_j^k) = 0 \\ \mathbf{y}_i^k = 1 &\rightarrow \alpha(\mathbf{y}_i^k, \mathbf{x}_j^k) = 1 \vee \alpha(\mathbf{y}_i^k, \mathbf{x}_j^k) = 2 \end{aligned} \qquad \textbf{1.47}$$

de acuerdo 4.47 es evidente que los mínimos valores de las componentes de la $k-ésima$ matriz, están sólo en la $k-ésima$ fila, incluyen sólo los valores 0 y 1, y dependen exclusivamente de los valores en las componentes de $\mathbf{x}^k$, es decir, cuando $\mathbf{x}_j^k = 1 \leftrightarrow \alpha(\mathbf{y}_k^k, \mathbf{x}_j^k) = 0$ o $\mathbf{x}_j^k = 0 \leftrightarrow \alpha(\mathbf{y}_k^k, \mathbf{x}_j^k) = 1$. Por lo tanto, dado que las asociaciones del conjunto fundamental son creadas de tal manera que a un patrón de entrada le corresponde *uno y sólo uno* de los patrones de salida y, además, el valor de $k \in \{1,2,...p\}$ fue tomado de manera arbitraria, se puede afirmar que la matriz $\mathbf{\Lambda}$ es afectada en su $i-ésima$ fila por el patrón $\mathbf{x}^i$ y lo hace con U_i valores 0 y $(n-U_i)$ valores 1. Por último, podemos reescribir la fase de aprendizaje con la siguiente expresión: $\forall i \in \{1,2,...,p\}$ y $\forall j \in \{1,2,...,n\}$

$$\lambda_{ij} = \alpha(\mathbf{y}_i^i, \mathbf{x}_j^i) \qquad \textbf{1.48}$$

Lema 5: Sea $\mathbf{r}$ el vector suma min, de la definición 6, de la memoria heteroasociativa $\mathbf{\Lambda}$, entonces, $r_i = C_i \quad \forall i \in \{1,2,...p\}$

Demostración: Sea $\mathbf{r}$ el vector suma min de la memoria heteroasociativa $\mathbf{\Lambda}$ expresado, de acuerdo a la definición 6, como sigue:

$$r_i = \sum_{j=1}^{n} \sim T_j \qquad \textbf{1.49}$$

donde $T \in B^n$ y las componentes de T se definen así:

$$T_j = \begin{cases} 0 & \leftrightarrow \quad \lambda_{ij} = 0 \\ 1 & \leftrightarrow \quad \lambda_{ij} \neq 0 \end{cases} \qquad \textbf{1.50}$$

$\forall j \in \{1,2,...,n\}$.
Por otro lado, sabemos por la definición 5 que:

$$C_h = \sum_{j=1}^{n} \sim x_j^h \qquad \textbf{1.51}$$

Particularmente

$$C_i = \sum_{j=1}^{n} \sim x_j^i \qquad \textbf{1.52}$$

Además sabemos por el lema 4 en su expresión *4.48* que $\mathbf{x}^i$ afecta a la matriz $\mathbf{\Lambda}$ sólo en su $i-ésima$ fila, por lo tanto es posible expresar a *4.49* de la siguiente manera:

$$r_i = \sum_{j=1}^{n} \sim \alpha(y_{i,}^i x_j^i) \qquad \textbf{1.53}$$

y dada la forma en que fue construido el vector $\mathbf{y}$ sabemos que $y_i^i = 0, \forall i \in \{1,2,...,p\}$ y por lo que la expresión $\alpha(y_{i,}^i x_j^i)$ depende únicamente de x_j^i y, de acuerdo con el lema 1

$$r_i = \sum_{j=1}^{n} \sim x_j^i \qquad \textbf{1.54}$$

por último por transitividad de las expresiones *4.51* y *4.54* concluimos que:

$$r_i = \left(\sum_{j=1}^{n} x_j^i \right) = C_i \qquad \textbf{1.55}$$

Lema 6: Sea $\mathbf{\Lambda}$ una memoria heteroasociativa tipo *Min* cuyo conjunto fundamental está expresado como sigue $\{(\mathbf{x}^\mu, \mathbf{y}^\mu) | \ \mu = 1,2,...p\}$ y $\mathbf{x}^\varpi \in A^n, A \in \{0,1\}$ el patrón fundamental que será presentado a la memoria $\mathbf{\Lambda}$. Después de la fase de recuperación del algoritmo

original de memoria heteroasociativa Alfa-Beta tipo *Max* se obtendrá un vector columna $\mathbf{z}^{\varpi} \in A^p$ el cual presenta el valor 0 en cada componente cuyo índice i es el índice de la fila de la memoria $\mathbf{\Lambda}$ que corresponde a los patrones fundamentales mayores o iguales a $\mathbf{x}^{\varpi}$, es decir:

$$\forall i \quad z_i^{\varpi} = 0 \rightarrow x^i \geq x^{\omega}, x^i \in \left\{ \left(\mathbf{x}^{\mu}, \mathbf{y}^{\mu} \right) | \, \mu = 1,2,...p \right\} \qquad \textbf{1.56}$$

Demostración: Dada la fase de recuperación original de las memorias heteroasociativas tipo *Min*, sabemos que:

$$z_i^{\varpi} = 0 \rightarrow \bigvee_{j=1}^{n} \beta(\lambda_{ij}, x_j^{\varpi}) = 0 \qquad \textbf{1.57}$$

y para que $\bigvee_{j=1}^{n} \beta(\lambda_{ij}, x_j^{\varpi}) = 0$, como la función β sólo puede arrojar 0 o 1, es necesario que $\beta(\lambda_{ij}, x_j^{\varpi}) = 0. \forall j \in \{1,2,...,n\}$. Ahora, a partir de lo anterior y por lema 4 sólo se pueden dar los siguientes casos: $\forall j \in \{1,2,...,n\}$

$$\beta(\lambda_{ij}, x_j^{\varpi}) = 0 \rightarrow \begin{cases} \lambda_{ij} = 1 \wedge x_j^{\varpi} = 0 \\ \lambda_{ij} = 0 \wedge (x_j^{\varpi} = 1 \vee x_j^{\varpi} = 0) \end{cases} \qquad \textbf{1.58}$$

Ahora, dado que sabemos por el lema 4 que cada patrón x^i afecta sólo a la $i-ésima$ fila de la memoria $\mathbf{\Lambda}$ y además conocemos el funcionamiento del operador *alfa* de la fase de aprendizaje, podemos inferir de la expresión *4.58* lo siguiente:

$\forall j$ siempre que suceda $(\lambda_{ij} = 0 \wedge x_j^{\varpi} = 1)$ o $(\lambda_{ij} = 1 \wedge x_j^{\varpi} = 0)$ podemos concluir que $x^i = x^{\varpi}$; además, $\forall j$ siempre que suceda $(\lambda_{ij} = 0 \wedge x_j^{\varpi} = 1)$ o $\exists j(\lambda_{ij} = 0 \wedge x_j^{\varpi} = 0)$ podemos concluir que $x^i > x^{\varpi}$. Es decir

$$x^i \geq x^{\varpi} \qquad \textbf{1.59}$$

por lo tanto por transitividad de *4.57, 4.58 y 4.59* podemos concluir que:

$$z_i^{\varpi} \rightarrow x^i \geq x^{\varpi} \qquad \textbf{1.60}$$

Teorema 2: Sea $\mathbf{\Lambda}$ una memoria heteroasociativa tipo *Min* con su conjunto fundamental expresado por $\left\{ \left(\mathbf{x}^{\mu}, \mathbf{y}^{\mu} \right) | \, \mu = 1,2,...p \right\}$, construido sin ninguna asociación repetida. Sea $\mathbf{x}^{\varpi} \in A^n$ con $\varpi \in \{1,2,...,p\}$ un patrón fundamental, el cual a sido presentado como patrón de entrada a la matriz $\mathbf{\Lambda}$, y $\mathbf{z}^{\varpi} \in A^p$ el vector resultante de la fase original de recuperación de las memorias heteroasociativas tipo *Min* con $A \in \{0,1\}, p, n \in Z^+$. La fase de recuperación del nuevo modelo de memoria heteroasiciativa tipo *Min* siempre presentará recuperación correcta, esto es, con el modelo propuesto siempre obtendremos la correspondiente $\mathbf{y}^{\varpi}$ sin ambigüedad.

Demostración: El algoritmo de recuperación del nuevo modelo de memoria heteroasociativa tipo Min esta expresado de la siguiente manera:

$$\mathbf{y}_i^{\varpi} = \begin{cases} 0 & \text{si} \quad r_i = \bigwedge_{k\in\theta} r_k \quad \wedge \quad z_k^{\varpi} = 0 \\ 1 & en\ otro\ caso \end{cases} \qquad \textbf{1.61}$$

donde $\theta = \left\{ i \mid Z_i^{\varpi} = 0 \right\}$.

Para demostrar la recuperación correcta del nuevo algoritmo es necesario asegurarnos de que el componente r_i que corresponde al patrón correcto es el mínimo. Esto puede ser demostrado por *contradicción.*

Supongamos que $x^{\alpha} \in A^n$ y $x^{\beta} \in A^n$ son los patrones correspondientes a $z_{\alpha}^{\varpi} = 0$ y $z_{\beta}^{\varpi} = 0$ resultantes al presentarle un patrón $x^{\varpi} \in A^n$ a la memoria $\mathbf{\Lambda}$, siendo $x^{\alpha}, x^{\beta}, x^{\varpi} \in \left\{ \left(\mathbf{x}^{\mu}, \mathbf{y}^{\mu} \right) \mid \mu = 1,2,...p \right\}$ con x^{α} el patrón correcto y x^{β} un patrón espurio arbitrario. Además se tiene para cada patrón su correspondiente valor r_i, particularmente r_{α} y r_{β} tal que $r_{\alpha} < r_{\beta}$. Ahora asumimos la negación de lo que queremos probar:

$$r_{\alpha} \geq r_{\beta} \qquad \textbf{1.62}$$

Ahora, de acuerdo al lema 5 sabemos que la expresión *4.62* puede ser expresada de la siguiente manera:

$$C_{\alpha} \geq C_{\beta} \qquad \textbf{1.63}$$

Ahora por el lema 6 sabemos que cada patrón espurio x^i, donde x^{ϖ} es el correcto implica $\forall i$, $z_i^{\varpi} = 0, \mathbf{x}^i \neq \mathbf{x}^{\varpi} \rightarrow \mathbf{x}^i > \mathbf{x}^{\varpi}$ por lo tanto podemos tomar por hipótesis que:

$$\mathbf{x}^{\beta} > \mathbf{x}^{\alpha} \qquad \textbf{1.64}$$

Lo cual por la definición 4 podemos expresar a *4.64* como

$$C_{\beta} > C_{\alpha} \qquad \textbf{1.65}$$

lo cual nos lleva a una contradicción entre *4.63* y *4.65* por lo tanto podemos afirmar que $C_{\alpha} \geq C_{\beta}$ no puede ser cierto por lo que $C_{\alpha} < C_{\beta}$ es verdadera. Dado que $\mathbf{x}^{\beta}$ fue tomada de manera arbitraria, esta condición es cierta para cada patrón espurio.

Ejemplo 8: Sean los vectores los patrones de entrada

$$x^1 = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}, x^2 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, x^3 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, x^4 = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 1 \end{pmatrix}$$

y sus correspondientes clases

$$y^1 = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 1 \end{pmatrix}, y^2 = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \end{pmatrix}, y^3 = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \end{pmatrix}, y^4 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \end{pmatrix}$$

Las clases están construidas de acuerdo a la codificación cero-hot y a cada clase le corresponde uno y sólo uno de los patrones de entrada. El conjunto fundamental queda de la siguiente manera:

$$\left\{\left(x^1, y^1\right), \left(x^2, y^2\right), \left(x^3, y^3\right), \left(x^4, y^4\right)\right\}$$

Paso 1:

$$\begin{pmatrix} 0 \\ 1 \\ 1 \\ 1 \end{pmatrix} \boxtimes \begin{pmatrix} 1 & 0 & 1 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 2 & 1 & 1 & 2 \\ 1 & 2 & 1 & 1 & 2 \\ 1 & 2 & 1 & 1 & 2 \end{pmatrix}$$

$$\begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \end{pmatrix} \boxtimes \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 2 & 1 & 2 & 2 & 2 \\ 1 & 0 & 1 & 1 & 1 \\ 2 & 1 & 2 & 2 & 2 \\ 2 & 1 & 2 & 2 & 2 \end{pmatrix}$$

$$\begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \end{pmatrix} \boxtimes \begin{pmatrix} 1 & 0 & 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 2 & 1 & 2 \\ 1 & 2 & 2 & 1 & 2 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 2 & 2 & 1 & 2 \end{pmatrix}$$

$$\begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \end{pmatrix} \boxtimes \begin{pmatrix} 1 & 1 & 0 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 2 & 1 & 1 \\ 1 & 1 & 2 & 1 & 1 \\ 1 & 1 & 2 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Se aplica el operador binario mínimo $\wedge$ a las matrices obtenidas en el paso anterior de las memorias tipo *Min*

$$\mathbf{\Lambda} = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Fase de Recuperación:

Al presentarle un patrón x^{ϖ} con $\varpi \in \{1,2,...,p\}$, particularmente x^3, a la memoria Λ obtenida en la fase de aprendizaje

$$\Lambda \nabla_\beta x^3 = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix} \nabla_\beta \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

De acuerdo con el paso 2 de la fase de recuperación del nuevo algoritmo de memorias heteroasociativas tipo *Min*, se obtiene el vector resultante se denomina z^4, y además valores de r

$$z^3 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, r = \begin{pmatrix} 3 \\ 1 \\ 2 \\ 4 \end{pmatrix}$$

Ahora, según la expresión del paso 2 de la fase de recuperación del nuevo algoritmo de memorias heteroasociativas tipo *Min*:

$$y^3 = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \end{pmatrix}$$

ya que el mínimo valor de r_j donde $z_j^3 = 0 \forall j \in \{1,2,3,4\}$ es 2.

Tanto para las nuevas Memorias Heteroasiciativas Alfa-Beta tipo *Max* y *Min* prevalece el hecho de que son muy sensibles al ruido mezclado. Para el caso de las nuevas Memorias Heteroasiciativas Alfa-Beta tipo *Max* sucede que son muy robustas al ruido aditivo y son muy sensibles a ruido sustractivo. Y en caso contrario para las nuevas Memorias Heteroasociativas Alfa-Beta tipo *Min* son muy robustas a ruido sustractivo pero muy sensible a ruido aditivo.

Una propuesta de solución para el tratamiento del ruido mezclado se propone a continuación.

1.2 Multimemorias Heteroasociativas Alfa-Beta

En memorias asociativas, particularmente las memorias asociativas Alfa-Beta, es común encontrar una correspondencia de uno a uno entre el conjunto fundamental y el número de memorias creadas a partir de éste; es decir, para cada conjunto fundamental distinto existe una y sólo una matriz de aprendizaje.

La creación de varias memorias asociativas, las cuales se comportarán como una sola en el sentido de que en la fase de recuperación se obtenga un solo vector de salida puede ser útil para el tratamiento del ruido mezclado.

Para el desarrollo de este algoritmo es necesario establecer las siguientes definiciones.

Definición 7: Sean $A=\{0,1\}$, $n,q\in Z^+$ y $q\neq 0$ y sea $\mathbf{x}^{\varpi}\in A^n$, $\varpi\in\{1,2,...,p\}$. Es posible dividir a $\mathbf{x}^{\varpi}$ en q particiones iguales si $\frac{n}{q}\in Z^+$.

Ejemplos 9: Si tomamos el vector x^1 del ejemplo 1

$$x^1=\begin{pmatrix}1\\0\\1\\1\\0\end{pmatrix}$$

Sólo existen dos formas de dividir el vector x^1 de tal manera que dichas particiones sean iguales ya que $\frac{n}{q}\in Z^+$ se cumple solo para dos casos.

$$\frac{5}{1}=5,\frac{5}{2}=2.5,\frac{5}{3}=1.66,\frac{5}{4}=1.25,\frac{5}{5}=1$$

Definición 8: Sea $A=\{0,1\},\mathbf{x}^{\alpha}\in A^n$, $\alpha\in\{1,2,...,p\}$ un vector columna y q el numero de particiones que se desean de dicho vector con $n,q,\in Z^+$. La operación de particionamiento vectorial ρ de $\mathbf{x}^{\alpha}$ se define como el conjunto de vectores columna binario $\frac{n}{q}-$dimensionales. Dicho particionamiento se denota por:

$$\rho(\mathbf{x}^{\alpha},q)=\left\{\mathbf{x}^{\alpha 1},\mathbf{x}^{\alpha 2},...,\mathbf{x}^{\alpha q}\right\} \qquad \textbf{1.66}$$

tal que $\mathbf{x}^{\alpha l}\in A^{\frac{n}{q}}$ con $l\in\{1,2,...,q\}$

Ejemplo 10: Aplique el operador de particionamiento vectorial x^1 para obtener una sola partición.

$$\rho(x^1,1) = \{x^{11}\}$$

por otro lado, si deseamos obtener 5 particiones operación de particionamiento queda así:

$$\rho(x^1,5) = \{x^{11}, x^{12}, x^{13}, x^{14}, x^{15}\}$$

1.2.1 Multimemoria Alfa-Beta Max

Sean $A = \{0,1\}, n, p \in Z^+$, $\mu \in \{1,2,...p\}, i \in \{1,2,...,p\}$ y $j \in \{1,2,...n\}$, y sean $\mathbf{x} \in A^n$ y $\mathbf{y} \in A^p$ los vectores de entrada y salida, respectivamente. A partir de estos vectores es posible construir el conjunto fundamental expresado por $\{(\mathbf{x}^\mu, \mathbf{y}^\mu) \mid \mu = 1,2,..., p\}$ de acuerdo con el nuevo algoritmo de memoria heteroasociativa Alfa-Beta tipo *Max*.

Fase de Aprendizaje

PASO 1

Una vez que se tiene el conjunto fundamental, para construir a partir de un solo conjunto fundamental varias matrices de aprendizaje, en lugar de solo una, es necesario dividir cada patrón de entrada $\mathbf{x}^\mu$ en q particiones, generando así q vectores de entrada $\frac{n}{q}$ – dimensionales y por lo tanto q memorias asociativas.

Para cada $\mu \in \{1,2,...p\}$ a partir de la pareja ordenada $(\mathbf{x}^\mu, \mathbf{y}^\mu)$ se aplica la *operación de particionamiento vectorial* a cada uno de los vectores $\mathbf{x}^\mu$ de las p parejas ordenadas. Por lo que el nuevo conjunto fundamental puede ser expresado como sigue:

$$\{(\rho(\mathbf{x}^\mu, q), \mathbf{y}^\mu) \mid \mu = 1,2,..., p\}$$

aplicando el operador de particionamiento el conjunto fundamental queda como sigue:

$$\{(\{\mathbf{x}^{\mu 1}, \mathbf{x}^{\mu 2},..., \mathbf{x}^{\mu q}\}, \mathbf{y}^\mu) \mid \mu = 1,2,..., p\}$$

y por distributividad

$$\{(\mathbf{x}^{\mu 1}, \mathbf{y}^\mu), (\mathbf{x}^{\mu 2}, \mathbf{y}^\mu),..., (\mathbf{x}^{\mu q}, \mathbf{y}^\mu) \mid \mu = 1,2,..., p\}$$

Ahora, para $l \in \{1,2,..., q\}$ a partir de las parejas $(\mathbf{x}^{\mu l}, \mathbf{y}^\mu)$ se construyen las q matrices según la fase de aprendizaje del nuevo algoritmo de memorias heteroasociativas presentado en esta tesis.

$$[\mathbf{y}^\mu \boxtimes (\mathbf{x}^{\mu l})^t]_{mx(\frac{n}{q})} \qquad \textbf{1.67}$$

Se aplica el operador binario máximo $\vee$ a las matrices obtenidas

$$\mathbf{V}^l = \bigvee_{\mu=1}^{p} [\mathbf{y}^\mu \boxtimes (\mathbf{x}^{\mu l})^t]_{mx(\frac{n}{q})} \quad \textbf{1.68}$$

del tal modo que la entrada $ij-ésima$ de cada una de las q matrices esté dada por

$$v_{ij}^l = \bigvee_{\mu=1}^{p} \alpha(y_i^\mu, x_j^{\mu l}) \quad \textbf{1.69}$$

Fase de Recuperación

Al igual que en la construcción de las matrices de aprendizaje, para la fase de recuperación, es necesario dividir en q particiones el patrón que se presenta a las q memorias.

PASO 1

Al presentarse un patrón $\mathbf{x}^\varpi$, con $\varpi \in \{1, 2, ...p\}$ a las $\mathbf{V}^l$ memorias, lo primero es aplicar el *operador de particionamiento* al vector $\mathbf{x}^\varpi$. De esta forma el vector quedará dividido en q nuevos vectores.

$$\rho(\mathbf{x}^\varpi, q) = \{\mathbf{x}^{\varpi 1}, \mathbf{x}^{\varpi 2}, ..., \mathbf{x}^{\varpi q}\} \quad \textbf{1.70}$$

Ahora, para cada una de las q matrices y q particiones del vector, se realiza la fase de recuperación del nuevo algoritmo de memorias heteroasociativas Alfa-Beta tipo *Max* propuesto en esta tesis y, se realiza para cada uno de ellos la operación Δ_β, asignando el resultado al vector $z^{\varpi l}$

$$z^{\varpi l} = \mathbf{V}^l \Delta_\beta \mathbf{x}^{\varpi l} \quad \textbf{1.71}$$

Por lo que la $i-ésima$ componente de cada uno de los vectores columna resultantes se expresa como:

$$z_i^{\varpi l} = \bigwedge_{j=1}^{n} \beta(v_{ij}^l, x_j^{\varpi l}) \quad \textbf{1.72}$$

posteriormente se continúa con el paso 2 de la fase de recuperación del nuevo algoritmo de memoria heteroasociativa tipo *Max* propuesto en esta tesis, para cada uno de los vectores $z^{\varpi l}$.

Una vez que se lleve a cabo este procedimiento se obtendrán q vectores resultantes, uno por cada una de las matrices de aprendizaje.

PASO 2

Una vez que se tienen los q vectores $z^{\varpi l}$. Para obtener un solo vector resultante se deberá llevar a cabo el siguiente algoritmo.

Se crea un vector intermedio I el cual contendrá la sumatoria de las $i-ésimas$ componentes de los vectores $z^{\varpi l}$.

$$I_i^{\varpi} = \sum_{l=1}^{q} z_i^{\varpi l} \qquad \textbf{1.73}$$

Una vez que se obtiene el vector I^{ϖ}, el vector y^{ϖ} se obtendrá mediante la siguiente expresión

$$y^{\varpi} = \begin{cases} 1 & I_i^{\varpi} = \bigvee_{k=1}^{p} I_k^{\varpi} \\ 0 & otro\ caso \end{cases} \qquad \textbf{1.74}$$

Ejemplo 11: Sean los patrones de entrada

$$x^1 = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{pmatrix}, x^2 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}, x^3 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}, x^4 = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

y sus correspondientes clases

$$y^1 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, y^2 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, y^3 = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, y^4 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

las clases están construidas de acuerdo a la codificación one-hot y a cada clase le corresponde uno y sólo uno de los patrón de entrada por lo que el conjunto fundamental queda de la siguiente manera:

$$\left\{\left(x^1, y^1\right), \left(x^2, y^2\right), \left(x^3, y^3\right), \left(x^4, y^4\right)\right\}$$

Fase de Aprendizaje:

De acuerdo con la fase de aprendizaje de las multimemorias Alfa-Beta *Max*, lo primero es dividir cada patrón de entrada en el número de particiones deseadas. Se verifica que sea posible dividirlo en 3 particiones.

$$\frac{6}{3} = 2$$

Dado que $2 \in Z^+$, es posible dividir cada vector en 3 partes iguales de dimensión 2.

Ahora aplicamos el operador de particionamiento vectorial:

$$\left\{\left(\rho(x^1,3), y^1\right), \left(\rho(x^2,3), y^2\right), \left(\rho(x^3,3), y^3\right), \left(\rho(x^4,3), y^4\right)\right\}$$

ó

$$\left\{\left(\left\{x^{11}, x^{12}, x^{13}\right\}, y^1\right), \left(\left\{x^{21}, x^{22}, x^{23}\right\}, y^2\right), \left(\left\{x^{31}, x^{32}, x^{33}\right\}, y^3\right), \left(\left\{x^{41}, x^{42}, x^{43}\right\}, y^4\right)\right\}$$

de manera particular cada partición queda como sigue

$$\begin{aligned}
\rho(x^1,3) &= \{1 \quad 0,1 \quad 1,0 \quad 0\} \\
\rho(x^2,3) &= \{0 \quad 1,0 \quad 0,0 \quad 1\} \\
\rho(x^3,3) &= \{1 \quad 0,0 \quad 1,0 \quad 1\} \\
\rho(x^4,3) &= \{1 \quad 1,0 \quad 1,1 \quad 1\}
\end{aligned}$$

por lo que ahora el conjunto fundamental es el siguiente:

$$\left\{\left\{\left(x^{11}, y^1\right), \left(x^{21}, y^2\right), \left(x^{31}, y^3\right), \left(x^{41}, y^4\right)\right\}, \left\{\left(x^{12}, y^1\right), \left(x^{22}, y^2\right), \left(x^{32}, y^3\right), \left(x^{42}, y^4\right)\right\}, \left\{\left(x^{13}, y^1\right), \left(x^{23}, y^2\right), \left(x^{33}, y^3\right), \left(x^{43}, y^4\right)\right\}\right\}$$

Ahora, a partir de las parejas $(\mathbf{x}^{\mu l}, \mathbf{y}^{\mu})$ se construyen las q matrices según la fase de aprendizaje. Las 3 matrices se muestran a continuación:

$$\mathbf{V}^1 = \begin{pmatrix} 1 & 2 \\ 2 & 1 \\ 1 & 2 \\ 1 & 1 \end{pmatrix}, \mathbf{V}^2 = \begin{pmatrix} 1 & 1 \\ 2 & 2 \\ 2 & 1 \\ 2 & 1 \end{pmatrix}, \mathbf{V}^3 = \begin{pmatrix} 2 & 2 \\ 2 & 1 \\ 2 & 1 \\ 1 & 1 \end{pmatrix}$$

Fase de Recuperación

Al presentarle un patrón x^{ϖ} con $\varpi \in \{1,2,...,p\}$, particularmente x^4, a las memorias $\mathbf{V}^l$ obtenidas en la fase de aprendizaje de las multimemorias Alfa-Beta Max. Se divide el patrón x^4 en el mismo número de particiones en que se dividieron los patrones en la fase de aprendizaje, en este caso, $q = 3$.

$$\rho(x^4,3) = \{1 \quad 1,0 \quad 1,1 \quad 1\}$$

una vez que se llevó a cabo la partición del vector, cada uno de los nuevos vectores, x^{41}, x^{42}, x^{43} es presentado con su correspondiente matriz de acuerdo con la fase de recuperación del nuevo algoritmo de memorias heteroasociativas Alfa-Beta tipo Max, por lo tanto se generan 3 vectores de salida, z^{41}, z^{42}, z^{43}, uno por cada partición.

$$z^{41} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}, z^{42} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \end{pmatrix}, z^{43} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

una vez formado cada vector se procede con el paso 2 de la fase de recuperación de las Multimemorias Alfa-Beta tipo *Max*. Primero que nada se crea el vector intermedio I^{ϖ}.

$$I^4 = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 3 \end{pmatrix}$$

Una vez que se obtiene el vector I^4, el vector y^4 se obtendrá mediante la siguiente función

$$y_i^{\varpi} = \begin{cases} 1 & I_i^{\varpi} = \bigvee_{k=0}^{n} I_k^{\varpi} \\ 0 & en\ otro\ caso \end{cases} \qquad \textbf{1.75}$$

por lo tanto:

$$y^4 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

1.2.2 Multimemoria Alfa-Beta Min

Sean $A=\{0,1\}$, $n, p \in Z^+$, $\mu \in \{1,2,...p\}, i \in \{1,2,...,p\}$ y $j \in \{1,2,...n\}$, y sean $\mathbf{x} \in A^n$ y $\mathbf{y} \in A^p$ vectores de entrada y salida, respectivamente. A partir de estos vectores es posible construir el conjunto fundamental expresado por $\{(\mathbf{x}^\mu, \mathbf{y}^\mu) \mid \mu = 1,2,...,p\}$ de acuerdo con el nuevo algoritmo de memoria heteroasociativa Alfa-Beta tipo *Min*.

Fase de Aprendizaje

PASO 1

Para construir a partir de un solo conjunto fundamental varias matrices de aprendizaje, en lugar de solo una, es necesario dividir cada patrón de entrada $\mathbf{x}^\mu$ en q particiones, generando así q vectores de entrada $\frac{n}{q}-$dimensionales y por lo tanto q memorias asociativas.

Para cada $\mu \in \{1,2,...p\}$ a partir de la pareja ordenada $(\mathbf{x}^\mu, \mathbf{y}^\mu)$ se aplica la *operación de particionamiento vectorial* a cada uno de los vectores $\mathbf{x}^\mu$ de las p parejas ordenadas. Por lo que el nuevo conjunto fundamental puede ser expresado como sigue:

$$\{(\rho(\mathbf{x}^\mu, q), \mathbf{y}^\mu) \mid \mu = 1,2,..., p\}$$

Aplicando el operador de particionamiento el conjunto fundamental queda como sigue:

$$\{(\{\mathbf{x}^{\mu 1}, \mathbf{x}^{\mu 2},..., \mathbf{x}^{\mu q}\}, \mathbf{y}^\mu) \mid \mu = 1,2,..., p\}$$

y por distributividad

$$\{(\mathbf{x}^{\mu 1}, \mathbf{y}^\mu), (\mathbf{x}^{\mu 2}, \mathbf{y}^\mu),..., (\mathbf{x}^{\mu q}, \mathbf{y}^\mu) \mid \mu = 1,2,..., p\}$$

Ahora, a partir de las parejas $(\mathbf{x}^{\mu l}, \mathbf{y}^\mu)$ se construyen las q matrices de acuerdo con la fase de aprendizaje del nuevo algoritmo presentado en esta tesis.

$$[\mathbf{y}^\mu \boxtimes (\mathbf{x}^{\mu l})^t]_{mx(\frac{n}{q})} \qquad \textbf{1.76}$$

Se aplica el operador binario máximo $\wedge$ a las matrices obtenidas

$$\mathbf{\Lambda}^l = \bigwedge_{\mu=1}^{p} [\mathbf{y}^\mu \boxtimes (\mathbf{x}^{\mu l})^t]_{mx(\frac{n}{q})} \qquad \textbf{1.77}$$

del tal modo que la entrada $ij-ésima$ de cada una de las q matrices esté dada por

$$\lambda_{ij}^{l} = \bigwedge_{\mu=1}^{p} \alpha(y_i^{\mu}, x_j^{\mu l}) \qquad \textbf{1.78}$$

Fase de Recuperación

Al igual que en la construcción de las matrices de aprendizaje, para la fase de recuperación es necesario dividir en q particiones el patrón que se presentará a las q memorias.

PASO 1

Al presentarse un patrón $\mathbf{x}^{\varpi}$, con $\varpi \in \{1, 2, ...p\}$ a la multimemoria heteroasociativa Alfa-Beta tipo *Min*, lo primero es aplicar el *operador de particionamiento* al vector $\mathbf{x}^{\varpi}$. De esta forma el vector quedará dividido en q nuevos vectores.

$$\rho(\mathbf{x}^{\varpi}, q) = \left\{\mathbf{x}^{\varpi 1}, \mathbf{x}^{\varpi 2}, ..., \mathbf{x}^{\varpi q}\right\} \qquad \textbf{1.79}$$

Ahora, para cada una de las q matrices y particiones del vector, se realiza la fase de recuperación del algoritmo de memorias heteroasociativas Alfa-Beta tipo *Min* propuesto en esta tesis, es decir, se realiza para cada uno de ellos la operación Δ_{β}, asignando el resultado a el vector $z^{\varpi l}$

$$z^{\varpi l} = \mathbf{\Lambda}^{l} \nabla_{\beta} \mathbf{x}^{\varpi l} \qquad \textbf{1.80}$$

Por lo que la $i-ésima$ componente de cada uno de los vectores columna resultantes, se expresa como:

$$z_i^{\varpi l} = \bigvee_{j=1}^{n} \beta(\lambda_{ij}^{l}, x_j^{\varpi l}) \qquad \textbf{1.81}$$

posteriormente se continúa con el paso 2 de la fase de recuperación del algoritmo de memoria heteroasociativa tipo *Min* propuesto en esta tesis, para cada $z^{\varpi l}$.

Una vez que se lleve a cabo este procedimiento se obtendrán q vectores resultantes, uno por cada una de las matrices de aprendizaje.

PASO 2

Una vez que se tienen los q vectores resultantes, a los que denominaremos $z^{\varpi l}$, para obtener un solo vector resultante se deberá llevar a cabo el siguiente algoritmo.

Se crea un vector intermedio I el cual contendrá la sumatoria de las $i-ésimas$ componentes de los vectores $z^{\varpi l}$.

$$I_i^{\varpi} = \sum_{l=1}^{q} z_i^{\varpi l} \qquad \textbf{1.82}$$

Una vez que se obtiene el vector I^{ϖ}, el vector y^{ϖ} se obtendrá mediante la siguiente función:

$$y_i^{\varpi} = \begin{cases} 0 & I_i^{\varpi} = \bigwedge_{k=1}^{p} I_k^{\varpi} \\ 1 & en\ otro\ caso \end{cases} \qquad \textbf{1.83}$$

***Ejemplo 12*:** Sean los patrones de entrada

$$x^1 = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{pmatrix}, x^2 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}, x^3 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}, x^4 = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

y sus correspondientes clases

$$y^1 = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 1 \end{pmatrix}, y^2 = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \end{pmatrix}, y^3 = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \end{pmatrix}, y^4 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \end{pmatrix}$$

las clases están construidas de acuerdo a la codificación cero-hot y a cada clase le corresponde uno y sólo uno de los patrones de entrada, por lo que el conjunto fundamental queda de la siguiente manera:

$$\left\{\left(x^1, y^1\right), \left(x^2, y^2\right), \left(x^3, y^3\right), \left(x^4, y^4\right)\right\}$$

De acuerdo con el paso 1 de la fase de aprendizaje lo primero verifica que sea posible dividir el vector en 3 particiones conforme a la definición 7.

$$\frac{6}{3} = 2$$

ya que $2 \in Z^+$, es posible dividir cada vector en 3 particiones iguales de dimensión 2.

Ahora aplicamos el operador de particionamiento vectorial:

$$\left\{\left(\rho(x^1,3), y^1\right), \left(\rho(x^2,3), y^2\right), \left(\rho(x^3,3), y^3\right), \left(\rho(x^4,3), y^4\right)\right\}$$

o

$$\left\{\left(\left\{x^{11}, x^{12}, x^{13}\right\}, y^1\right), \left(\left\{x^{21}, x^{22}, x^{23}\right\}, y^2\right), \left(\left\{x^{31}, x^{32}, x^{33}\right\}, y^3\right), \left(\left\{x^{41}, x^{42}, x^{43}\right\}, y^4\right)\right\}$$

de manera particular cada partición queda como sigue:

$$\rho(x^1,3) = \{1 \quad 0,1 \quad 1,0 \quad 0\}$$
$$\rho(x^2,3) = \{0 \quad 1,0 \quad 0,0 \quad 1\}$$
$$\rho(x^3,3) = \{1 \quad 0,0 \quad 1,0 \quad 1\}$$
$$\rho(x^4,3) = \{1 \quad 1,0 \quad 1,1 \quad 1\}$$

por lo que ahora el conjunto fundamental es el siguiente:

$$\{\{(x^{11}, y^1), (x^{21}, y^2), (x^{31}, y^3), (x^{41}, y^4)\}, \{(x^{12}, y^1), (x^{22}, y^2), (x^{32}, y^3), (x^{42}, y^4)\}, \{(x^{13}, y^1), (x^{23}, y^2), (x^{33}, y^3), (x^{43}, y^4)\}\}$$

A partir de las parejas $(\mathbf{x}^{\mu l}, \mathbf{y}^{\mu})$ se construyen las q matrices según la fase de aprendizaje

$$\mathbf{\Lambda}^l = \bigwedge_{\mu=1}^{p} [\mathbf{y}^{\mu} \boxtimes (\mathbf{x}^{\mu l})^t]_{mx(\frac{n}{q})} \qquad \textbf{1.84}$$

dichas matrices se muestran a continuación:

$$\mathbf{\Lambda}^1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix}, \mathbf{\Lambda}^2 = \begin{pmatrix} 0 & 0 \\ 1 & 1 \\ 1 & 0 \\ 1 & 0 \end{pmatrix}, \mathbf{\Lambda}^3 = \begin{pmatrix} 1 & 1 \\ 1 & 0 \\ 1 & 0 \\ 0 & 0 \end{pmatrix}$$

Fase de Recuperación

Al presentarle un patrón x^{ϖ} con $\varpi \in \{1, 2, ..., p\}$, particularmente x^3, a las memorias $\mathbf{\Lambda}^l$, de acuerdo con el paso 1, tenemos que dividir el patrón x^3 en el mismo número de particiones en que se dividieron los patrones en la fase de aprendizaje de las multimemorias heteroasociativas Alfa-Beta, en este caso, 3.

$$\rho(x^3, 3) = \{1 \quad 0,0 \quad 1,0 \quad 1\}$$

una vez que se llevó a cabo la partición del vector, cada uno de los nuevos vectores, x^{31}, x^{32}, x^{33} es presentado con su correspondiente matriz de acuerdo con la fase de recuperación del nuevo algoritmo de memorias heteroasociativas Alfa-Beta tipo Min, como consecuencia se generan 3 vectores de salida $z^{31}, z^{32}, z^{33},$ uno por cada partición.

$$z^{31} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}, z^{32} = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix}, z^{33} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

Posteriormente se crea el vector intermedio I^{ϖ} el cual contendrá la sumatoria de las $i-ésimas$ componentes de los vectores $z^{\varpi l}$.

$$I^4 = \begin{pmatrix} 2 \\ 2 \\ 0 \\ 2 \end{pmatrix}$$

Una vez que se obtiene el vector I^4 y los correspondientes vectores z^{31}, z^{32}, z^{33}, el vector y^4 resultante es el siguiente:

$$y^4 = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \end{pmatrix}$$

1.3 Clasificador Basado en Multimemorias Alfa-Beta

En esta sección se presenta el algoritmo de clasificación de patrones que toma como base las multimemorias heteroasociativas Alfa-Beta. Dicho algoritmo fue desarrollado principalmente para aplicaciones en bioinformática y se denominará de aquí en adelante como CMMAB (Clasificador MultiMemoria Alfa-Beta).

Primero que nada, dado que las Memorias Heteroasociativas Alfa-Beta trabajan sólo con patrones binarios es esencial codificar, en binario, cualquier instancia con las que vayamos a trabajar. En este caso, las instancias con las que trabajamos son secuencias de ADN, por lo que es necesario encontrar cierta codificación que represente a cada elemento de las secuencias. Para ello se creó una correspondencia entre los caracteres del ADN y patrones binarios. Después de un proceso intenso de búsqueda de la mejor codificación, resultó evidente, por los resultados encontrados en esta tesis, que la codificación *one-hot* es la óptima, hecho reportado en la referencia [89]. La correspondencia es la siguiente:

Nucleótido	Secuencia Binaria
A	1000
C	0100
T	0010
G	0001

Esta codificación es útil especialmente en situaciones como la siguiente: cuando sucede que en algunas posiciones de la secuencia de ADN que vayamos a codificar es necesario indicar más de un posible valor para dicha posición. Por ejemplo, si en esa posición el valor puede ser “A” o “T”, la codificación para ese nuevo elemento consistirá sólo en hacer una operación *OR* entre la codificación de “A” con la de “T”. Esto es posible dadas las características de cada una de las memorias, ya sea *Max* o *Min,* de soportar cierto tipo de alteraciones en los patrones a recuperar.

1.3.1 Algoritmo de Clasificación

Fase de Aprendizaje

1. Se toma *x* número de elementos por cada clase (el valor de *x* puede variar de clase en clase) para crear el conjunto fundamental. Siendo los primeros *x* elementos del conjunto fundamental los correspondientes la primera clase, los segundos *x* elementos a la segunda clase y así sucesivamente hasta tomar elementos de todas las clases.
2. Se crea la matriz de aprendizaje de acuerdo con la fase de aprendizaje del algoritmo de Multimemorias Heteroasociativas Alfa-Beta *Max*.
3. Se crea la matriz de aprendizaje de acuerdo con la fase de aprendizaje del algoritmo de Multimemorias Heteroasociativas Alfa-Beta *Min*.

Fase de Recuperación

1. Se presenta el patrón que se desea clasificar a la matriz de aprendizaje de la Multimemorias Heteroasociativas Alfa-Beta *Max* y se almacena el vector resultante.
2. Se presenta el patrón que se desea clasificar a la matriz de aprendizaje de la Multimemorias Heteroasociativas Alfa-Beta *Min* y se almacena el vector resultante.
3. Se realiza un OR lógico entre el vector resultante de la memoria tipo *Max* y la negación del vector resultante de la memorias tipo *Min* de tal forma que los componentes en los que al menos uno de los vectores tenga un *1* prevalezcan en el tercer vector al que llamaremos *vector clase total*.
4. Dado que se conoce el número de elementos que se tomaron por cada clase para formar el conjunto fundamental. Se suman los primeros *x* elementos del vector *clase*, los cuales corresponden a la primera clase; se suman los segundos *x* elementos del mismo *vector clase total* los cuales corresponden a la segunda clase y así sucesivamente hasta terminar las clases y tener tantas sumatorias como clases.
5. Para determinar a cual de las clases corresponde el patrón presentado a las matrices se escoge, como la clase correcta, a la(s) mayor(es) sumatoria(s).
6. En caso de que exista más de una sumatoria igual, no será posible determinar a cuál de las clases pertenece y se considerar como clasificación errónea.

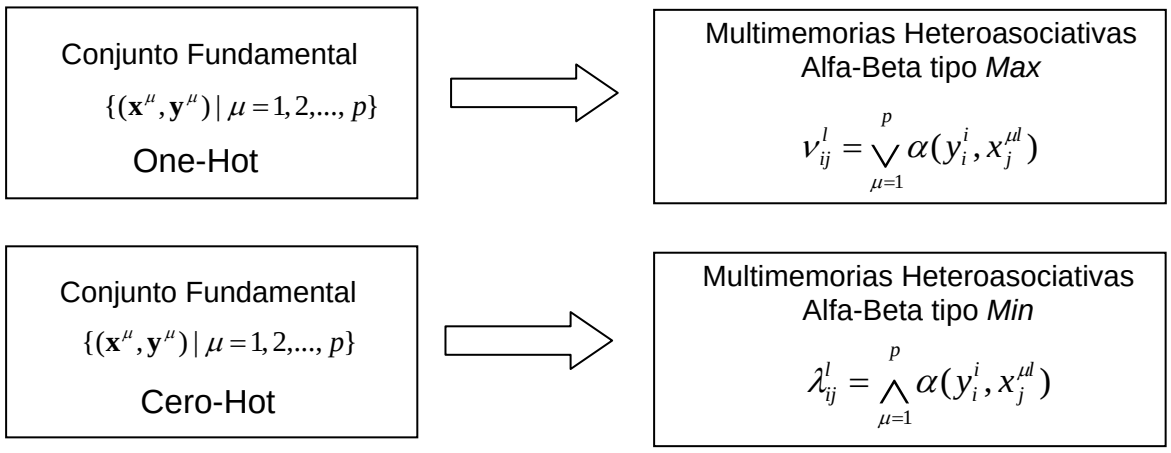


Figura 1.1 Fase de Aprendizaje del Clasificador

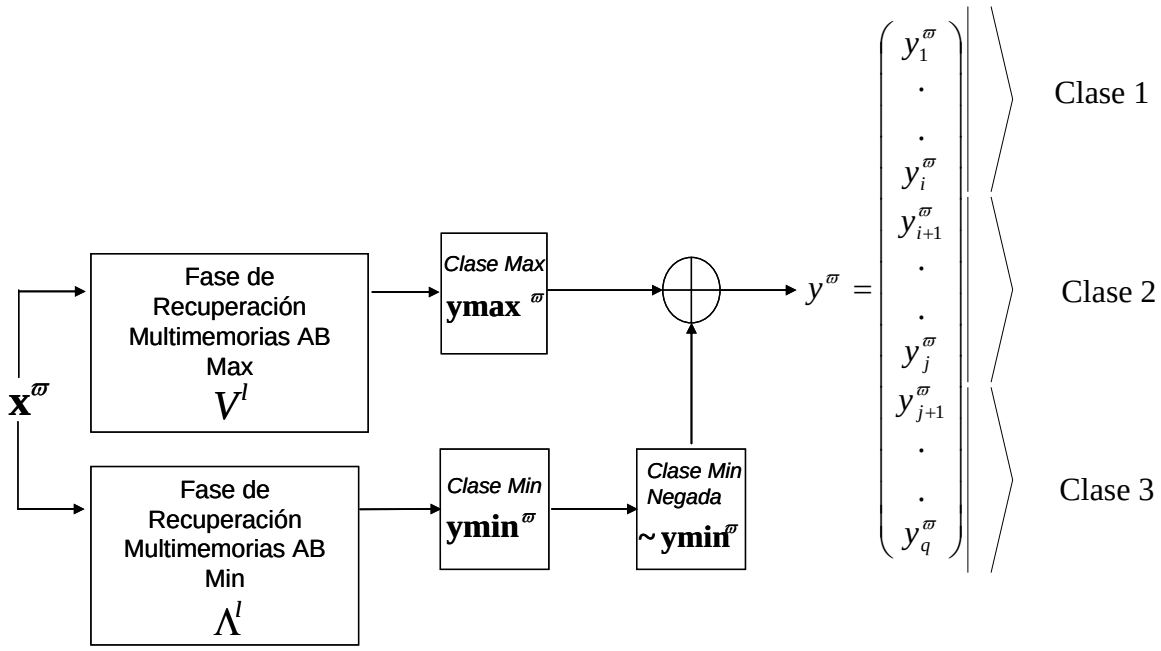


Figura 1.2 Fase de Recuperación del Clasificador

Ejemplo 13: Sean los patrones de entrada divididos en clases como se indica a continuación

$$x^1 = \begin{pmatrix}1\\0\\1\\1\\0\\0\end{pmatrix}, x^2 = \begin{pmatrix}0\\1\\0\\0\\0\\1\end{pmatrix}, x^3 = \begin{pmatrix}1\\0\\0\\1\\0\\1\end{pmatrix}, x^4 = \begin{pmatrix}1\\1\\0\\1\\1\\1\end{pmatrix}, x^5 = \begin{pmatrix}1\\0\\0\\1\\1\\1\end{pmatrix}, x^6 = \begin{pmatrix}1\\1\\1\\0\\0\\0\end{pmatrix}$$

Clase 1 Clase 2 Clase 3

y las correspondientes clases, tanto para la memoria tipo *Max, one-hot*, como la *Min, cero-hot*.

$$y^1 = \begin{pmatrix}1\\0\\0\\0\\0\\0\end{pmatrix}, y^2 = \begin{pmatrix}0\\1\\0\\0\\0\\0\end{pmatrix}, y^3 = \begin{pmatrix}0\\0\\1\\0\\0\\0\end{pmatrix}, y^4 = \begin{pmatrix}0\\0\\0\\1\\0\\0\end{pmatrix}, y^5 = \begin{pmatrix}0\\0\\0\\0\\1\\0\end{pmatrix}, y^6 = \begin{pmatrix}0\\0\\0\\0\\0\\1\end{pmatrix}$$

$$yc^1=\begin{pmatrix}0\\1\\1\\1\\1\\1\end{pmatrix}, yc^2=\begin{pmatrix}1\\0\\1\\1\\1\\1\end{pmatrix}, yc^3=\begin{pmatrix}1\\1\\0\\1\\1\\1\end{pmatrix}, yc^4=\begin{pmatrix}1\\1\\1\\0\\1\\1\end{pmatrix} yc^5=\begin{pmatrix}1\\1\\1\\1\\0\\1\end{pmatrix}, yc^6=\begin{pmatrix}1\\1\\1\\1\\1\\0\end{pmatrix}$$

El conjunto fundamental para cada una de las memorias queda como sigue:

$$\left\{(x^1,y^1),(x^2,y^2),(x^3,y^3),(x^4,y^4),(x^5,y^5),(x^6,y^6)\right\}$$

$$\left\{(x^1,yc^1),(x^2,yc^2),(x^3,yc^3),(x^4,yc^4),(x^5,yc^5),(x^6,yc^6)\right\}$$

Una vez que se tienen los dos conjuntos fundamentales es necesario llevar la partición sobre cada uno de los conjuntos fundamentales, para este caso q=3, con lo que los nuevos conjuntos fundamentales, al aplicar el operador de particionamiento quedan de la siguiente manera:

Se verifica que sea posible el particionamiento:

$$\frac{n}{q}=\frac{6}{3}=2$$

dado que $2\in Z^+$ entonces es posible aplicar el operador de particionamiento vectorial, por lo tanto:

$$\left\{\left(\left\{x^{11},x^{12},x^{13}\right\},y^1\right),\left(\left\{x^{21},x^{22},x^{23}\right\},y^2\right),\left(\left\{x^{31},x^{32},x^{33}\right\},y^3\right),\left(\left\{x^{41},x^{42},x^{43}\right\},y^4\right),\left(\left\{x^{51},x^{52},x^{53}\right\},y^5\right),\left(\left\{x^{61},x^{62},x^{63}\right\},y^6\right)\right\}$$

$$\left\{\left(\left\{x^{11},x^{12},x^{13}\right\},yc^1\right),\left(\left\{x^{21},x^{22},x^{23}\right\},yc^2\right),\left(\left\{x^{31},x^{32},x^{33}\right\},yc^3\right),\left(\left\{x^{41},x^{42},x^{43}\right\},yc^4\right),\left(\left\{x^{51},x^{52},x^{53}\right\},yc^5\right),\left(\left\{x^{61},x^{62},x^{63}\right\},yc^6\right)\right\}$$

Una vez que se conoce el conjunto fundamental se crean las correspondientes memorias, tanto las *Max* como las *Min*.

Max

$$\mathbf{V}^1=\begin{pmatrix}1&2\\2&1\\1&2\\1&1\\1&2\\1&1\end{pmatrix}, \mathbf{V}^2=\begin{pmatrix}1&1\\2&2\\2&1\\2&1\\2&1\\1&2\end{pmatrix}, \mathbf{V}^3=\begin{pmatrix}2&2\\2&1\\2&1\\1&1\\1&1\\2&2\end{pmatrix}$$

Min

$$\boldsymbol{\Lambda}^1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix}, \boldsymbol{\Lambda}^2 = \begin{pmatrix} 0 & 0 \\ 1 & 2 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}, \boldsymbol{\Lambda}^3 = \begin{pmatrix} 1 & 1 \\ 1 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 1 \end{pmatrix}$$

Hasta este punto se llevó a cabo la fase de aprendizaje del nuevo clasificador. La fase de recuperación se ejemplifica a continuación.

Supongamos que se desea clasificar el patrón x^4:

$$x^4 = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

dicho patrón es presentado a cada una de las memorias de acuerdo con la fase de recuperación de la multimemoras heteroasociativas Alfa-Beta, obteniendo los correspondientes vectores:

Fase de Recuperación AB Max

$$y\max = (\mathbf{V}^l \Delta_\beta x^{4l}) = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

Fase de Recuperación AB Max

$$y\min = (\boldsymbol{\Lambda}^l \nabla_\beta x^{4l}) = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \end{pmatrix}$$

Una vez que se tienen los dos vectores es necesario negar el vector ymin:

$$\sim y\min = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

Por último se lleva a cabo la operación lógica OR entre *ymax* y *ymin* y se obtiene la clase a la que pertenece el patrón

$$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} OR \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

Clase 1

Clase 2

Clase 3

CAPÍTULO 5. Resultados y Discusión

El presente capítulo es una muestra de cómo la modificación en los algoritmos, tanto de aprendizaje como de recuperación de las memorias heteroasociativas Alfa-Beta en conjunto con el algoritmo multimemorias heteroasociativas Alfa-Beta, permiten la creación de un algoritmo de notable desempeño para clasificación de patrones (CMMAB), particularmente en el área de bioinformática.

Los experimentos se realizaron en una Laptop HP, con un procesador AMD Turion a 1.60 GHz, 512 MByte de RAM y 60 GBytes en disco duro. Se usó el sistema operativo Microsoft Windows XP Profesional y un software, codificado con C++ Builder 6, hecho especialmente para la prueba de bases de datos de bioinformática.

El funcionamiento del software utilizado en este trabajo de tesis se describe y ejemplifica en el apéndice B.

1.1 Descripción de las Bases de Datos

En la literatura científica podemos encontrar varios autores que ocupan ciertas bases de datos para probar sus algoritmos de clasificación. En este caso nos enfocaremos sobre dos de las más comúnmente usadas: “E. coli promoter gene sequences (DNA) with associated imperfect domain theory” y “Primate splice-junction gene sequences (DNA) with associated imperfect domain theory”; fueron tomadas del sitio web de la Universidad de California en Irvine [88] y ambas están relacionadas con el problema de clasificación de secuencias de ADN.

El primero de los experimentos que se realizan en este trabajo de tesis es la separación de secuencias en dos clases principales, promotores y no-promotores. Para ello se utiliza la primera de las dos bases de datos anteriores: “E. coli promoter gene sequences (DNA) with associated imperfect domain theory”, la cual fue diseñada a partir de dos bases de datos, la primera de promotores creada por C. Harley y R. Raynolds y la segunda de no-promotores hecha por Tom Record [81].

Las características de esta base de datos son las siguientes:

- 106 instancias; cada una de éstas está constituida por 3 atributos: *clase*, *nombre* y *secuencia genética*.
- La *clase* identifica si la secuencia se trata de un promotor (+) o no (-), el *nombre* es un identificador para cada secuencia genética y por último la *secuencia genética*.
- La secuencia genética está conformada por 57 elementos o nucleótidos y cada uno de ellos puede tomar un valor de entre los siguientes 4: {A,T,G,C.}.
- La distribución de las clases es del 50%; es decir, 53 instancias son consideradas como promotores (+) y las otras 53 como no-promotores (-).

La segunda bases de datos utilizada en los experimentos de esta tesis, es conocida como "Primate splice-junction gene sequences (DNA) with associated imperfect domain theory". Esta base de datos contiene secuencias genéticas representativas de dos clases distintas, las cuales son: exon-intron (EI) e intron-exon (IE); el resto de los registros que no pertenecen a ninguna de las dos clases de interés, se pueden agrupar en una tercera clase que corresponde a secuencias no significativas ya sea como unión exon-intron o como unión intron-exon.

Las características de esta base de datos son las siguientes:

- 3190 instancias; cada una de éstas está constituida por 3 atributos: *clase*, *nombre* y *secuencia genética*.
- La *clase* identifica si la secuencia se trata de un empalme exon-intron (EI) o de un empalme intron-exon (IE), considerando que el resto de los datos no pertenecen a ninguno de estos dos casos; el *nombre* es un identificador para cada secuencia genética y por último la *secuencia genética*.
- La secuencia genética está conformada por 60 elementos o nucleótidos y cada uno de ellos puede tomar un valor de entre los siguientes 8: {A,T,G,C,D,N,S,R}. La Tabla 1.1 detalla cada uno de los valores en binario.
- La distribución de las clases es: 25% (767) para EI, 25% (768) para IE y 50% (1655) para los restantes que no pertenecen a ninguna de las dos clases de interés.

Tabla 1.1 Codificación binaria de Nucleótidos

Nucleótido	Secuencia Binaria
A	1000
C	0100
T	0010
G	0001
D	1011
N	1111
S	0101
R	1001

1.2 Reportes de Resultados

1.2.1 Clasificación de promotores en secuencias de ADN

En el caso de clasificación de secuencias de promotores se cuenta con dos artículos: J. Ortega [82] y Baffes & Money [87]. Es posible comparar al principal resultado de esta tesis, el clasificador CMMAB, con dichos artículos ya que se realizaron los experimentos bajo las mismas condiciones. Se toman, aleatoriamente, 85 instancias para formar el conjunto de entrenamiento, y a las 21 instancias restantes para el conjunto de experimentación. Se realizan 100 iteraciones y se reporta el mejor resultado, como el porcentaje de aciertos. A continuación se muestra una tabla comparativa de los resultados de dichos algoritmos contra los resultados del CMMAB.

Tabla 1.2 Resultados Experimentales

Conjunto de Entrenamiento	Ortega [82]	Baffes & Mooney [87]	CMMAB
85/106	92.5%	93%	98%
80/106	N/D	N/D	98%
70/106	N/D	N/D	96%

donde N/D es No Determinado

Es evidente que CMMAB supera sin problema a las dos propuestas de [82] y [87].

Al notar que, bajo las mismas condiciones, nuestra propuesta presenta resultados superiores a los de los otros dos trabajos, se decidió variar dichas condiciones disminuyendo el número de instancias tomadas para el conjunto de entrenamiento del algoritmo, y dejando un número mayor de instancias experimentales. El objetivo de esto es hacer notar que con menos elementos de aprendizaje, con nuestra propuesta, es posible obtener igual o mejores porcentajes que los presentados por los otros dos trabajos.

Las condiciones en cada uno de los experimentos son las siguientes:

- Se toman tres cantidades distintas de instancias de entrenamiento; 85, 80, 70 respectivamente, dejando las restantes como instancias experimentales.
- Para cada uno de los casos se llevaron a cabo 100 iteraciones, y en cada una de ellos las instancias de entrenamiento fueron tomadas aleatoriamente.
- Se presentaron al clasificador tanto las instancias de aprendizaje como las de experimentación.
- Se consideraron los 20 mejores resultados para cada uno de los casos. Para estos 20 se determinaron los siguientes valores: *porcentaje máximo*, *porcentaje mínimo y porcentaje promedio*.

La Tabla 1.3 muestra tres casos distintos en los que se varía el número de instancias tomadas para el entrenamiento del clasificador. Además se presentan las gráficas para cada uno de los casos.

Tabla 1.3 Comparativa variando el número de instancias para el conjunto de entrenamiento

Rendimiento del CMMAB	85 patrones de Entrenamiento	80 patrones de entrenamiento	70 patrones de entrenamiento
Mejor	98%	98%	96%
Peor	97%	96%	93%
Promedio	97.45%	96.7%	94.55%

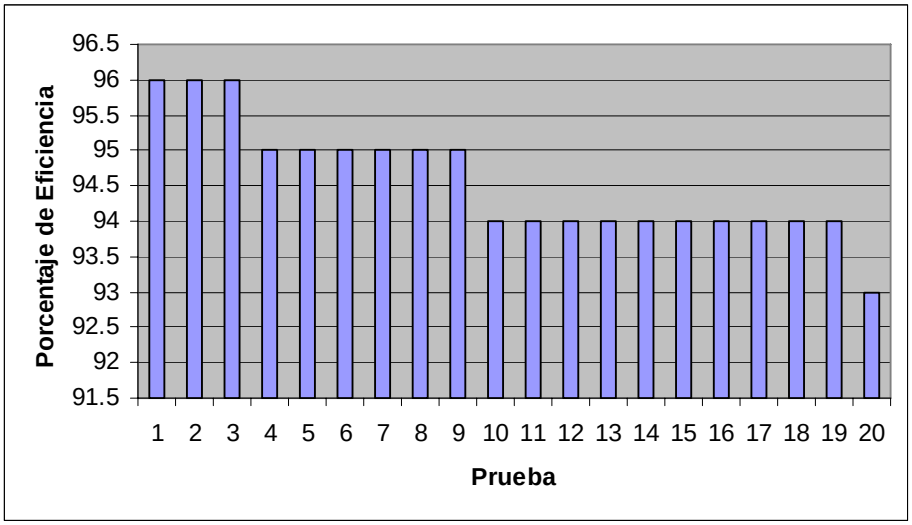


Figura 1.1 CMMAB entrenado con 70 patrones**.**

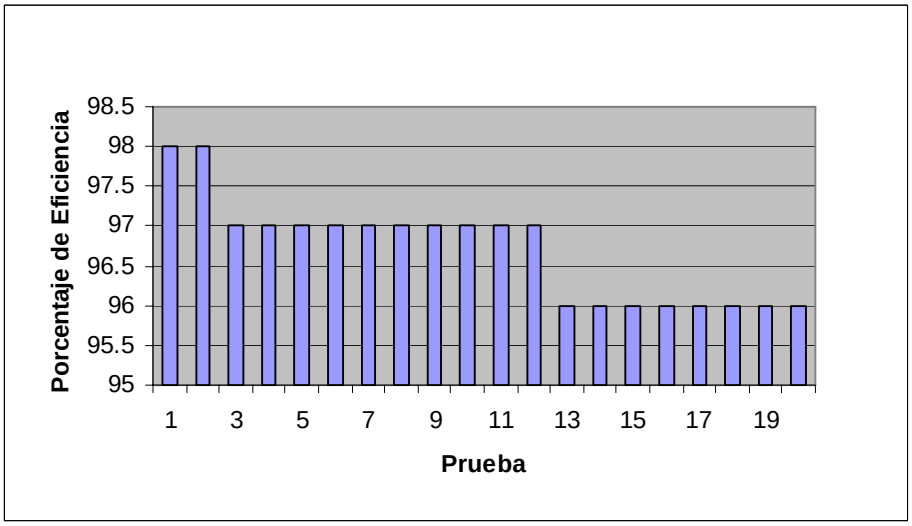


Figura 1.2 CMMAB entrenado con 80 patrones.

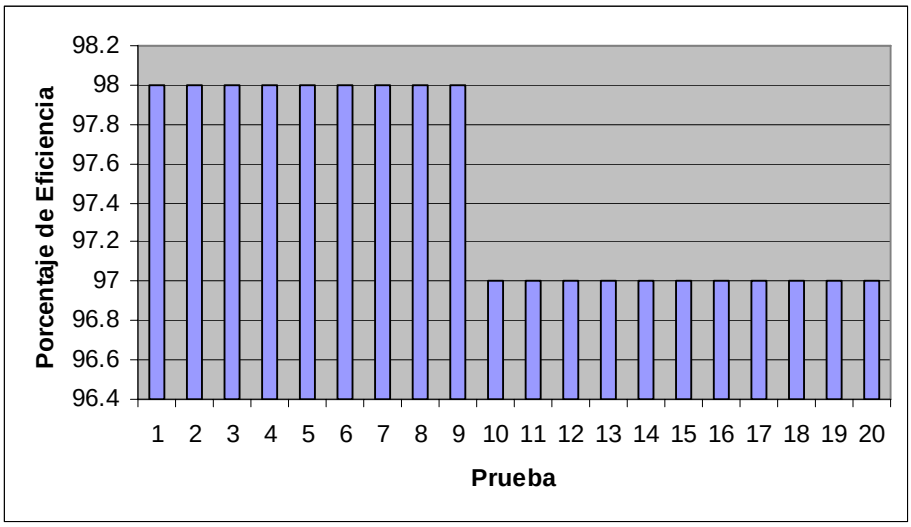


Figura 1.3 CMMAB entrenado con 85 patrones.

1.2.2 Clasificación de zonas de empalme en secuencias de ADN

En la clasificación de zonas de empalme (splice-junction), el clasificador CMMAB se compara con los resultados expuestos en el artículo de Rampone, S. [89]. En dicho artículo se presentan dos tipos de experimentos cuyas condiciones y resultados se exponen a continuación

Hold-Out

Se tomó, aleatoriamente, una muestra de 2000 instancias para el conjunto de entrenamiento y 1186 para el conjunto de prueba. En la tabla 5.4, que se muestra a continuación, aparecen los siguientes datos tomados de la referencia [89]: el número de elementos de cada una de las clases, el número de patrones clasificados erróneamente y la tasa de error.

Tabla 1.4 Comportamiento de BRAIN utilizando la metodología Hold-Out.

Clase	Número de instancias de entrenamiento (2000)	Número de instancias de entrenamiento (1186)	Número de Errores	Tasa de Error
EI	464	303	41/1186	3.4
IE	485	280	59/1186	4.9

Para probar el clasificador CMMAB se toma una muestra aleatoria de 2000 instancias para crear el conjunto de entrenamiento y se dejaron las restantes 1190 para el conjunto de prueba, y se realizaron 20 iteraciones reportando el valor más alto de aciertos. La razón por la que el número de instancias en el conjunto de prueba es distinto, es que en [89] se eliminaron algunos de los elementos repetidos de la base de datos.

Tabla 1.5 Comportamiento de CMMAB utilizando la metodología Hold-Out.

Clase	Número de instancias de entrenamiento (2000)	Número de instancias de entrenamiento (1190)	Número de Errores	Tasa de Error
EI	464	303	38/1190	3.1
IE	485	280	43/1190	3.6

En las Tabla 1.4 Tabla 1.5 se puede observar el mejor desempeño del CMMAB sobre el BRAIN para las clases de interés IE y. EI. No se incluyen datos relacionados con el resto de los registros de la base de datos, dado que no pertenecen a ninguna de las dos clases de interés.

Cross-Validation

Los resultados mostrados en el artículo [89] fueron obtenidos aplicando la metodología "10-Fold Cross-Validation" sobre una muestra de 1000 instancias tomadas aleatoriamente del total de 3190 instancias de la base de datos.

Usando la misma metodología, se realizaron experimentos con el clasificador CMMAB propuesto en esta tesis. La Tabla 1.6 muestra el desempeño de nuestro clasificador CMMAB respecto de los resultados mostrados por el algoritmo BRAIN y otros algoritmos de *Machine learning* consultados y presentados por Rampone, autor de [89].

Tabla 1.6 Desempeño del clasificador CMMAB respecto de otros 8 algoritmos clasificadores de patrones, usando 10-Fold Cross-Validation.

Sistema	EI	IE
BRAIN	0.050	0.040
CMMAB	0.068	0.078
KBANN	0.076	0.085
BackProp	0.057	0.107
PEBLS	0.082	0.075
Perceptron	0.163	0.174
ID3	0.106	0.140
COBWEB	0.150	0.095
Near. Neighbor	0.116	0.091

El algoritmo CMMAB, original de esta tesis, supera a todos los algoritmos, con excepción del BRAIN, en la clasificación de las clases de interés IE y EI. Obsérvese que el desempeño del CMMAB es superior a 7 de los 8 algoritmos con que se compara; el caso del BRAIN se explica por que este algoritmo fue creado precisamente para trabajar en la clasificación de los registros de la base de datos "Primate splice-junction gene sequences (DNA) with associated imperfect domain theory", y esto significa que BRAIN está sintonizado con esta base de datos, lo cual no sucede con el CMMAB, dado que es un algoritmo general de clasificación de patrones. Considerando lo anterior se propone, como trabajo a futuro, el diseño de algoritmos de similitud que permitan auxiliar en la clasificación de dichas instancias.

CAPÍTULO 6. Conclusiones y Trabajo Futuro

En este capítulo se presentan las conclusiones que se desprenden de este tema de tesis, además, se hacen propuestas sobre posibles trabajos a futuro, basados en las aportaciones de este trabajo.

1.1 Conclusiones

1. Se crearon nuevas herramientas en el área de las Memorias Asociativas.

2. Se extiende el modelo teórico original de las Memorias Heteroasociativas Alfa-Beta, de modo que permite la recuperación correcta de todos los patrones fundamentales.

3. Se presentan 8 definiciones 6 lemas y 2 teoremas que aseguran la recuperación correcta de los patrones fundamentales con el nuevo algoritmo de Memorias Heteroasociativas Alfa-Beta.

4. Se presenta un nuevo algoritmo de Multimemorias Heteroasociativa Alfa-Beta con el cual de diseña un algoritmo de clasificación de patrones que permite resolver problemas en bioinformática.

5. Experimentalmente se muestra que con el nuevo clasificador se obtienen mejores resultados que otros clasificadores, en problemas de Bioinformática, bajo las mismas condiciones.

1.2 Trabajo futuro

1. Aplicar el nuevo algoritmo de clasificación en otras bases de datos de Bioinformática [88].

2. Aplicar el nuevo algoritmo de clasificación para la predicción de estructuras secundarias en proteínas [58].

3. Búsqueda de nuevas tareas, diferentes a la clasificación, sobre las cuales se pueda aplicar el nuevo algoritmo de Memorias Heteroasociativas en Bioinformática [50,51].

4. Aplicar el nuevo algoritmo de clasificación a tareas no exclusivas de la bioinformática.

5. Desarrollar modelos de similitud que ayuden en la toma de decisiones en la clasificación de patrones en Bioinformática.

Desarrollar un nuevo modelo que permita la asignación de varios patrones a una sola clase.

Referencias

[1] Hassoun, M. H. 1993, *Associative Neural Memories*, Oxford University Press, New York.

[2] Kohonen, T. 1989, *Self-Organization and Associative Memory*, Springer-Verlag, Berlin.

[3] McCulloch, W. & Pitts, W. 1943, "A logical calculus of the ideas immanent in nervous activity", *Bulletin of Mathematical Biophysics*, vol. 5, pp. 115-133.

[4] Ritter, G. X. & Sussner, P. 1996, "An Introduction to Morphological Neural Networks" in *Proceedings of the 13th International Conference on Pattern Recognition*, vol. IV, Track D, pp. 709-717.

[5] Rosenblatt, F. 1958, "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain", *Psychological Review*, vol. 65, no. 6, pp. 386-408.

[6] Widrow, B. & Lehr M. A. 1990, "30 Years of Adaptive Neural Networks: Perceptron, Madaline and Backpropagation", *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1415-1441.

[7] Werbos, P. J. 1990, "Backpropagation Through Time: What It Does and How to Do It", *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550-1560.

[8] Hopfield, J.J. 1982, "Neural networks and physical systems with emergent collective computational abilities", *Proceedings of the National Academy of Sciences*, vol. 79, pp. 2554-2558.

[9] Minsky, M. & Papert, S. 1969, *Perceptrons*, MIT Press ,Cambridge.

[10] Steinbuch, K. 1961, "Die Lernmatrix", *Kybernetik*, vol. 1, no. 1, pp. 36-45.

[11 Willshaw, D., Buneman, O. & Longuet-Higgins, H. 1969, "Non-holographic associative memory", *Nature*, no. 222, pp. 960-962.

[12] Anderson, J. A. 1972, "A simple neural network generating an interactive memory", *Mathematical Biosciences*, vol. 14, pp. 197-220.

[13] Kohonen, T. 1972, "Correlation matrix memories", *IEEE Transactions on Computers, C-21*, vol. 4, pp. 353-359.

[14] Kosko, B. (1988), "Bidirectional associative memories", *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 18, no. 1, pp. 49-60.

[15] Haines, K. & Hecht-Nielsen, R., (1988), "A BAM With Increased Information Storage Capacity", *IEEE International Conference on Neural Networks*, vol. 1, pp. 181-190

[16] Leung, C.S. & Cheung, K.F., (1991), "Householder encoding for discrete bidirectional associative memory**",** *IEEE International Joint Conference on Neural Networks*, Vol. 1, pp. 237-241.

[17] Wang, Y.-F., Cruz J.B., Jr. & Mulligan, Jr., (1991, "Guaranteed recall of all training pairs for bidirectional associative memory", *IEEE Transactions on Neural Networks*, Vol. 1, Issue 6, pp. 559-567

[18] Shen, D. & Cruz, J.B., Jr., 2005, "Encoding strategy for maximum noise tolerance bidirectional associative memory", *IEEE Transactions on Neural Networks*, Vol. 16, Issue 2, pp. 293-300

[19] Ritter, G. X., Sussner, P. & Diaz-de-Leon, J. L. 1998, "Morphological associative memories", *IEEE Transactions on Neural Networks*, vol. 9, pp. 281-293.

[20] Yánez, C. 2002, *Memorias Asociativas basadas en Relaciones de Orden y Operadores Binarios*, Tesis de Doctorado, Centro de Investigación en Computación, México.

[21] Acevedo-Mosqueda, M.E. (2006), *Memorias Asociativas Bidireccionales Alfa-Beta*, Tesis de Doctorado en Ciencias de la Computación, Centro de Investigación en Computación, México.

[22] Acevedo-Mosqueda, M.E., Yáñez-Márquez, C.**,** & López-Yáñez, I. (2006). *A New Model of BAM: Alpha-Beta Bidirectional Associative Memories*, Lecture Notes in Computer Science, LNCS 4263, Springer-Verlag Berlin Heidelberg pp. 286-295. ISSN: 0302-9743.

[23] Acevedo-Mosqueda, M.E., Yáñez-Márquez, C. & López-Yáñez, I. (2006). *Alpha-Beta Bidirectional Associative Memories Based Translator*, IJCSNS International Journal of Computer Science and Network Security, Vol.6, No.5A, pp. 190-194. ISSN: 1738-7906.

[24] Marqués de Sá, J. P. (2001). Pattern Recognition, Concepts, Methods and Application, Germany, Springer.

[25] Duda, R. O., Hart, P. E., Stork, D. G. (2001) Pattern Classification, USA: Jhon Wiley & Sons.

[26] Shalkoff, R. (1992). Pattern Recognition, Statistical, Structural and Neural Approaches, USA: Jhon Wiley.

[27] Amari, S. (1977). Neural theory of associations and concept-formation. Biological Cybernetic, 26, 175-185.

[28] Anderson, J. A. & Rosenfeld, E. (Eds.) (1990). Neurocomputing: Foundation of Research, Cambridge: MIT Press.

[29] Steinbuch, K. & Frank, H. (1961), "Nichtdigitale Lernmatrizen als Perzeptoren", *Kybernetik*, vol. 1, no.3, pp. 117-124.

[30] Papadomanolakis, K., Kakarountas, A., Sklavos, N. & Goutis C. E., (2002), A Fast Johnson-Mobius Encoding Scheme for Fault Secure Binary Counters, *Proceedings of Design, Autamtions and Test in Europe*, (DATE '02), pp. 1-7.

[31] Yáñez Márquez, C. & Díaz de León Santiago, J.L. (2001). *Lernmatrix de Steinbuch*, IT-48, Serie Verde, ISBN 970-18-6688-6, CIC-IPN, México

[32] Abu-Mostafa, Y. & St. Jacques, J. (1985), "Information capacity of the Hopfield model", *IEEE Transactions on Information Theory*, IT-31, no. 4, pp. 461-464.

[33] Austin, J. (1987), "ADAM: A Distributed Associative Memory for Scene Analysis", In *Proceedings of First International Conference on Neural Networks*, pp. 285-295.

[34] Yáñez Márquez, C. & Díaz de León Santiago, J.L. (2001). *Memorias Morfológicas Autoasociativas*, IT-58, Serie Verde, ISBN 970-18-6698-3, CIC-IPN, México.

[35] Díaz de León Santiago, J.L. & Yáñez Márquez, C. (2001). *Memorias Morfológicas Heteroasociativas*, IT-57, Serie Verde, ISBN 970-18-6697-5, CIC-IPN, México.

[36] Anderson, J. A. & Rosenfeld, E. (1990). *Neurocomputing: Fundations of Research*, MIT Press ,Cambridge.

[37] Rosen, K., (1999), *Discrete Mathematics and Its Applications*, McGraw-Hill, Estados Unidos.

[38] Simpson, P. K. (1990), *Artificial Neural Systems*, Pergamon Press , New York.

[39] Ritter, G. X., Diaz-de-Leon, J. L. & Sussner, P. (1999). Morphological bidirectional associative memories, Neural Networks, 12, 851-867.

[40] Santiago-Montero, Raúl. (2003), Clasificador Hibrido de patrones basado en la Lernmatrix de Steinbuch y el Linear Associator de Anderson-Kohonen, Tesis de Maestría en Ciencias de la Computación, Centro de Investigación en Computación, México.

[41] Israel Roman-Godinez, Itzama Lopez-Yanez, Cornelio Yanez-Marquez, "A New Classifier Based on Associative Memories," *CIC*, pp. 55-59, 15th International Conference on Computing (CIC'06), 2006.

[42] Sossa, H., Barrón, R. & Vázquez, R. (2004), "New Associative Memories to Recall Real-Valued Patterns", *CIARP*, LNCS 3287, pp. 195-202.

[43] Baldi, P., and Brunak, S.: "Bioinformatics: the machine learning approach" (MIT Press, Cambridge, MA 1998).

[44] Doolittle, R. F. (1987). Of URFs and ORFs: A primer on how to analyze derived amino acid sequences, University Science Books, Mill Valley, California.

[45] Von Heijne, G. (1987). Sequence analysis in molecular biology: Treasure trove or trivial pursuit, Academic Press, London.

[46] Wolfsberg, T. G., Wetterstrand, K. A., Guyer, M. S., Collins, F. S. & Baxevanis, A. D. (2002). A user's guide to the human genome. Nature Genetics 32 (suppl).

[47] Altman, R.B., Valencia, A., Miyano, S. and Ranganathan, S.: Challenges for intelligent system in biology, IEEE Intell. Syst, 2001, 16,(6), pp 14-20.

[48] Setubal, J., and Meidanis, J.: Introduction to computational molecular biology (International Thomson Publishing, Boston, MA, 1999).

[49] S.S. Ray, S. Bandyopadhyay, P. Mitra and S.K. Pal.: Bioinformatics in neurocomputing framework. IEEE, Proceedings, 2005.

[50] Lesk, Arthur M.: Introduction to Bioinformatics, Oxford University Press, Second Edition, 2005.

[51] Mitra, S. Hayashi, Y. Bioinformatics with soft computing, Systems, Man and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on, pp. 616-635, 2006.

[52] Nash, H., Blair, D., and Grefenstette, J.: Comparing algorithms for large scale sequence analysis. Proc. 2nd IEEE Int. Symp. on Bioinformatics and Bioengineering.

[53] Needleman, S.B., and Wunsch, C.D: A general method applicable to the search for similarities in the amino acid sequence of two proteins, J. Mol. Biol., 1970, 48, pp. 443-453.

[54] Smith, T.F., and Waterman, M.S.: Identification of common molecular sequence, J. Mol. Biol., 1981, 147, pp. 195-197.

[55] Gautheret, D., Major, F. and Cedergren, R.: Pattern searching/ alignment with RNA primary and secondary structure: an effective descriptor for tRNA, Comp. Appl. Biosci., 1990, 6, pp. 325-331.

[56] Fickett, J.W.: 'Finding genes by computer: the state of the art', Trends Genet., 1996, 12, (8), pp. 316–320

[57] Salzberg, S.L., Searls, D.B., and Kasif, S.: 'Computational methods in molecular biology' (Elsevier Science, Amsterdam, 1998)

[58] Chou, P., and Fasmann, G.: 'Prediction of the secondary structure of proteins from their amino acid sequence', Adv. Enzym., 1978, 47, pp. 145–148.

[59] Luscombe, N.M., Greenbaum, D., and Gerstein, M.: 'What is bioinformatics? A proposed definition and overview of the field',Methods Informat. Med., 2001, 40, (4), pp. 346–358.

[60] Quackenbush, J.: 'Computational analysis of microarray data', Nat. Rev. Genetics, 2001, 2, pp. 418–427.

[61] Arakawa, M., Hasegawa, K., and Funatsu, K.: 'Application of the novel molecular alignment method using the Hopfield neural network to 3D-QSAR', J. Chem. Inf. Comput. Sci., 2003, 43, (5), pp. 1396–1402

[62] Jones, D.T.: 'GenTHREADER: an efficient and reliable protein fold recognition method for genomic sequences', J. Mol. Biol., 1999, 287, pp. 797–815

[63] Lukashin, A.V., Anshelevich, V.V., Amirikyan, B.R., Gragerov, A.I., and Frank-Kamenetskii, M.D.: 'Neural network models for promoter recognition', J. Biomol. Struct. Dyn., 1989, 6, (6), pp. 1123–1133

[64] Sun, J., Song, W.Y., Zhu, L.H., and Chen, R.S.: 'Analysis of tRNA gene sequences by neural network', J. Comput. Biol., 1995, 2, (3), pp. 409–416

[65] Kalate, R.N., Tambe, S.S., and Kulkarni, B.D.: 'Artificial neural networks for prediction of mycobacterial promoter sequences', Comput. Biol. Chem., 2003, 27, (6), pp. 555–564

[66] Qian, N., and Sejnowski, T.J.: 'Predicting the secondary structure of globular proteins using neural network models', J. Mol. Biol., 1988, 202, (4), pp. 865–884

[67] Rost, B., and Sander, C.: 'Improved prediction of protein secondary structure by use of sequence profiles and neural networks', Proc. Nat. Acad. Sci., 1993, 90, (16), pp. 7558–7562

[68] Rost, B., and Sander, C.: 'Prediction of protein secondary structure at better than 70% accuracy', J. Mol. Biol., 1993, 232, pp. 584–599

[69] Wood, M.J., and Hirst, J.D.: 'Predicting protein secondary structure by cascade-correlation neural networks', Bioinformatics, 2004, 20, (3), pp. 419–420

[70] Pasquier, C., Promponas, V.J., and Hamodrakas, S.J.: 'PREDCLASS: cascading neural networks for generalized protein classification and genome-wide applications', Proteins, 2001, 44, (3), pp. 361–369

[71] Raibaud O., and M. Schwartz M. Positive control of transcription initiation in bacteria. Annu. Rev. Genet. 1984, 18, 173-206.

[72] Gralla J.D., Promoter Recognition and mRNA initiation by Escherichia coli E sigma70. in Methods in Enzymology. 1990, 185, 37-54.

[73] Snyder L. and Champness W. Molecular Genetic of Bacteria. ASM Press Washington, D.C.

ISBN 1-55581-102-7.

[74] Mulligan M.E., Hawley D.K., Entriken R., and McClure W. Escherichia coli promoters sequence predict in vitro RNA polymerase selectivity. Nucleic Acids Res. 1984, 12, 789-800.

[75] Weller K., and Recknagel R.D., Promoters strength prediction based on occurrence frequencies of consensus patterns. J. theor. Biol., 1994, 171(4), 335-359.

[76] Lewin, B., Genes VI. Oxford University Press. 1997, 287-332.

[77] Reeder, R.H. Enhancers and ribosomal gene spacers. Cell Sep., 1984, 38(2):349-51.

[78] Moss T., and Stefanovsky V.Y., Promotion and regulation of ribosomal transcription in eukaryotes by RNA polymerase. Prog. Nucleic Acid Res. Mol. Biol., 1995, 50:25-66.

[79] Kahl B.F., Li H. and Paule M.R. DNA melting and promoter clearance by eukaryotic RNA polymerase I. J. Mol. Biol. 2000 ;299(1):75-89.

[80] Hawley D. K., and McClure W. R. The effect of a lambda repressor mutation on the activation of transcription initiation from the lambda PRM promoter. Cell Feb, 1983, 32(2):327-33.

[81] Calvin B.Harley and Robert P. Reynolds. Analysis of E. coli promoter sequences, January 26, 1987. Vol. 15, Num 5, 2343-2361.

[82] Ortega, J. (1995). *On the informativeness of the DNA promoter sequences domain theory*. Journal of Artificial Intelligence Research, 2, 361--367

[83] Geoffrey Towell, Jude Shavlik, and Michiel Noordewier. *Refinement of approximate domain theories by knowledge-based artificial neural networks*. In Proceedings of the Eighth National Conference on Artificial Intelligence, Boston, Massachusetts, 1990. MIT Press.

[84] Jacques Cohen. Bioinformatics—An Introduction for Computer Scientists. ACM Computing Surveys, Vol. 36, No. 2, June 2004, pp. 122–158.

[85] A. Tsakonas, T. Tsiligianni & G. Dounias. Evolutionary Neural Logic Networks in Splice-Junction Gene Sequences Classification. International Journal on Artificial Intelligence Tools Vol. 15, No. 2 (2006) 287–307.

[86] A. Lumini, L. Nanni. Identifying splice-junction sequences by hierarchical multiclassifier. Pattern Recognition Letters 27 (2006) 1390–1396.

[87] Baffes, P. T., & Mooney, R. J. (1993). Symbolic revision of theories with M-of-N rules. In Proceedings of the Thirteenth International Joint Conference on Artifcial Intelligence. Chambery, France.

[88] UCIDB, Irvine California. www.ics.**uci**.edu/~mlearn/ML**Repository**.html.

[89] Brunak, S., Engelbrecht, J, and Knudsen, S (1991) Prediction of human mRNA donor and acceptor sites from the DNA sequence. J. Mol. Biol. 220, 49-65.

Referencias

Glosario

Los conceptos que a continuación se presentan fueron extraídos de las siguientes referencias: [20], [21], [48],[49],[50],[51],[56],[59],[87],[89]

Ácido Desoxirribonucleico (ADN)

Constituye el material genético de los organismos. Es el componente químico primario de los cromosomas y el material del que los genes están formados.

Ácido Nucleico

Una gran molécula compuesta de subunidades de nucleótidos.

Ácido Ribonucleico de Transferencia (tRNA)

Es un tipo de RNA que tiene estructura con secuencias de tripletes de nucleótidos que son complementarias a las secuencias de los tripletes de nucleotidos del mRNA. La función del tRNA en la síntesis de proteínas es enlazar los aminoácidos y transportarlos a los ribosomas, donde las proteínas son ensambladas de acuerdo al código genético portado por el mRNA.

Ácido Ribonucleico Mensajero (mRNA)

El RNA que sirve como plantilla para la síntesis de proteínas.

Ácido Ribonucleico Polimerasa (RNA polimerasa)

Enzimas que catalizan la síntesis de ácidos nucleicos sobre plantillas de ácidos nucleicos preexistentes, ensamblando el RNA a partir de los ribonucleótidos o el ADN a partir de desoxiribonucleotidos.

Adenina

Una base nitrogenada, es un miembro del par de bases A-T (Adenina-Timina). ADN (ácido desoxirribonucleico). La molécula que contiene codificada la información

genética. El ADN es una molécula enrolada en forma de hélice, que se mantiene unida entre sí por medio de enlaces dobles entre los pares de bases o nucleotidos. Las cuatro bases que contiene los nucleotidos en el ADN son: adenina (A), guanina (G), citocina (C) y Timina (T). En la naturaleza, las pares de bases se forman solamente entre A y T y entre G y C, por lo tanto la secuencia de bases de cada una de las cadenas puede deducirse a partir de uno de sus pares.

Alineamiento de Secuencias

Es una forma de mostrar DNA, RNA, o estructuras primarias proteicas para resaltar las zonas de similitud, que podrían indicar relaciones funcionales o evolutivas entre los genes o proteínas consultados. Las secuencias alineadas se escriben con las letras (representando aminoácidos o nucleótidos) en columnas en las que se insertan espacios para que las zonas con idéntica o similar estructura se alineen.

Aminoácidos

Una cadena de compuestos proteicos de al menos 20 bases nitrogenadas, que se combinan para formar una proteína. La secuencia de aminoácidos en una proteína están determinados por la acción del código genético.

Backpropagation

Es un algoritmo de aprendizaje supervisado que se usa para entrenar redes neuronales artificiales. El algoritmo consiste en minimizar un error (comúnmente cuadrático) por medio de gradiente descendiente, por lo que la parte esencial del algoritmo es cálculo de las derivadas parciales de dicho error con respecto a los parámetros de la red neuronal.

Bioinformática

Es la aplicación de los ordenadores y los métodos informáticos en el análisis de datos experimentales y simulación de los sistemas biológicos.

Biología Molecular

Es el estudio de la vida a un nivel molecular. Esta área se solapa con otros campos de la Biología y la Química, particularmente Genética y Bioquímica. La biología molecular concierne principalmente al entendimiento de las interacciones de los diferentes sistemas de la célula, lo que incluye muchísimas relaciones, entre ellas las del ADN con

el RNA, la síntesis de proteínas, el metabolismo, y el cómo todas esas interacciones son reguladas para conseguir un afinado funcionamiento de la célula.

BLAST

(*Basic Local Alignment Search Tool*) es un programa informático de alineamiento de secuencias de tipo local ya sea de ADN o de proteínas. El programa es capaz de comparar una secuencia problema (comúnmente llamada query) contra una gran cantidad de secuencias que se encuentren en una base de datos. El algoritmo encuentra las secuencias de la base de datos que tienen mayor parecido a la secuencia Quero

Cadena polipeptídica

Los péptidos están formados por la unión de aminoácidos mediante un enlace peptídico.

Carbono

El carbono es un elemento químico de número atómico 6 y símbolo C. Es sólido a temperatura ambiente. Dependiendo de las condiciones de formación, puede encontrarse en la naturaleza en distintas formas alotrópicas, carbono amorfo y cristalino en forma de grafito o diamante. Es el pilar básico de la química orgánica; se conocen cerca de 10 millones de compuestos de carbono, y forma parte de todos los seres vivos conocidos.

Célula

En biología, la célula es la unidad más esencial que tiene todo ser vivo. Es además la estructura funcional fundamental de la materia viva según niveles de organización biológica, capaz de vivir independientemente como entidades unicelular, o bien, formar parte de una organización mayor, como un organismo pluricelular. La célula presenta dos modelos básicos: la procarionte y eucarionte. Su organización general comprende: membrana plasmática, citoplasma y ADN.

Citosina

Una base nitrogenada, un miembro del par de bases G-C (guanina y citosina).

Clase

Son los grupos o conjuntos de patrones que representan un mismo tipo de concepto

Clasificador

El objetivo de un clasificador es agrupar patrones con base en un conocimiento a priori o información estadística extraída de los patrones. Los patrones a clasificar suelen ser grupos de medidas u observaciones, definiendo puntos en un espacio multidimensional apropiado.

Cloroplasto

Los cloroplastos son los orgánulos en donde se realiza la fotosíntesis en las células vegetales y de los otros organismos fotosintetizadores. Están formados por un sistema de membranas interno en donde se encuentran ubicados los sitios en que se realiza cada una de las partes del proceso fotosintético.

Codificación

Proporcionar la secuencia de nucleótidos adecuada para la síntesis de una determinada proteína.

Codificación one-hot

La codificación one-hot funciona de la siguiente manera: para representar la clase $k \in \{1, 2, ..., p\}$, se asignan a las componentes del vector de salida y^{μ} los siguientes valores: $y_k^{\mu} = 1$ y $y_j^{\mu} = 0$ para j = 1, 2, ..., k-1, k+1, ..., p.

Código Genético

El código genético es la regla de correspondencia entre la serie de nucleótidos en que se basan los ácidos nucleicos y las series de aminoácidos (polipéptidos) en que se basan las proteínas. Es como el diccionario que permite traducir la información genética a estructura de proteína. A, T, G, y C son las "letras" del código genético y representan las bases nitrogenadas adenina, timina, guanina y citosina, respectivamente. Cada una de estas bases forma, junto con un glúcido (pentosa) y un grupo fosfato, un nucleótido; el ADN y el RNA son polímeros formados por nucleótidos encadenados.
Cada tres nucleótidos de la cadena (cada triplete) forman una unidad funcional llamada codón.

Codones

La información genética, contenida en el NRNA, se escribe a partir de cuatro letras, que corresponden a las bases nitrogenadas del RNA (A, C, G y U), las cuales van agrupadas de tres en tres. Cada grupo de tres se llama codón y lo que hace es codificar un aminoácido o un símbolo de puntuación (Comienzo, parada).

Conjunto fundamental

Es el conjunto finito de patrones $\{(x^{\omega},y^{\omega})|\omega \in \{1,2,\ldots p\}\}$ a partir del cual se diseña una memoria asociativa M, donde $x^{\omega} \in A$, $y^{\omega} \in A^{m}$, $p \in N$, y n y m son números naturales que representan las dimensiones de los patrones de entrada y salida, respectivamente. El conjunto A es de donde toman valores las componentes de los patrones de entrada x^{ω}, y es escogido arbitrariamente por el diseñador de la memoria M. El número p indica la cardinalidad del conjunto fundamental.

Contenido Genético

La secuencia de nucleótidos, codificada en tripletes (codones) a lo largo del mRNA, que determina la secuencia de aminoácidos en la síntesis de proteínas. La secuencia de ADN de un gen puede usarse para predecir la secuencia de mRNA, el código genético entonces puede usarse para predecir la secuencia de aminoácidos.

Cromosomas

La estructura genética con capacidad de autoreplicación de células conteniendo el ADN celular que porta en su nucleótido un arreglo de genes en forma de secuencia.

En procariontes, el ADN cromosomal es de tipo circular y todo el genoma es portado por un cromosoma. En eucariontes, los genomas consisten de un número de cromosomas cuyo ADN está asociado con diferentes tipos de proteínas.

Desoxirribosa

Su fórmula es $C5H10O4$. Ésta contiene toda la información genética que será transferida así de generación en generación. Por todo esto la desoxirribosa tiene una gran importancia en todo ser vivo existente. La información genética no se transfiere en la desoxirribosa pero sí es una parte fundamental de todo proceso de información genética ya que de éste se derivará la Ribosa

E. coli

Bacteria común que se ha estudiado intensamente por genetistas debido al tamaño pequeño de su genoma, baja patogenicidad y fácil crecimiento en el laboratorio.

Enlace Peptídico

Véase Cadena Polipeptídica.

Enlaces de Hidrógeno

Se produce un enlace de hidrógeno o puente de hidrógeno (correctamente llamado enlace por puente de hidrógeno) cuando un átomo de hidrógeno se encuentra entre dos átomos más electronegativos, estableciendo un vínculo entre ellos

Estructura Primaria de Proteínas

Se puede decir que la estructura primaria de las proteínas no es más que el orden de aminoácidos que la conforman.

Estructura Secundaria de Proteínas

Es el plegamiento que la cadena polipeptídica adopta gracias a la formación de enlaces de hidrógeno entre los átomos que forman el enlace peptídico. Los puentes de hidrógeno se establecen entre los estables.

Estructura Terciaria de Proteínas

Es el modo en que esa cadena polipeptídica se pliega en el espacio, es decir, a cómo se arrolla una determinada proteína globular. Es la disposición de los dominios en el espacio.
La estructura terciaria se realiza de manera que los aminoácidos apolares se sitúan hacia el interior y los polares hacia el exterior.

Eucariontes

Célula u organismo con membrana limitada, estructuralmente núcleo discreto y otros compartimentos subcelulares bien definidos. Los eucariontes incluyen todos los organismos excepto virus, bacterias y algas azul-verdosas. Compárese con procariontes.

Exones

Las secuencias de ADN de una proteína codificante de un gen, o de otra manera son las secuencias que están representadas en el mRNA en el momento que llega a los ribosomas, porque son las regiones del gen que se expresan. Comparar con intrones. Exonucleasa. Una enzima que penetra o rompe nucleótidos secuencialmente a partir de terminaciones libres de una línea de ácido nucleico sustraído.

Expresión Genética

El proceso por el cual la información codificada de un gen es convertida dentro de la estructura presente y operando en la célula. Los genes expresados incluyen aquellos que se transcriben dentro del mRNA y luego trasladados dentro de proteínas y aquellos que se transcriben dentro de RNA pero que no son trasladados como proteínas (por ej. los RNAs de transferencia y ribosomal).

Factor de Transcripción

Un factor de transcripción es una proteína que participa en la iniciación de la transcripción del ADN, pero que no forma parte de la RNA polimerasa. Los factores de transcripción actúan reconociendo sitios en el ADN o a otro factor, o a la RNA polimerasa.

Los factores de transcripción son proteínas que, tras ser estimuladas por señales citoplasmáticas, tienen la capacidad de regular la expresión génica en el núcleo celular.

Los factores de transcripción pueden ser activados o desactivados selectivamente por otras proteínas, a menudo como paso final de la cadena de transmisión de señales intracelulares.

Filogenéticos

Filogenia (del griego: phylon = tribu, raza y genetikos = relativo al nacimiento, de génesis = nacimiento) es la disciplina que estudia las relaciones evolutivas entre las distintas especies, reconstruyendo la historia de su diversificación (filogénesis) desde el origen de la vida en la Tierra hasta la actualidad. La filogenia proporciona el fundamento para la clasificación de los organismos.

Fosfatos

El fosfato forma parte de los nucleótidos, los monómeros en que se basa la composición del ADN y demás ácidos nucleicos. También hay fosfato en la composición de algunos

lípidos formadores de membranas, como los fosfoglicéridos, donde su elevada constante de ionización constribuye a la carga eléctrica de la «cabeza hidrófila».

Genes

La unidad física y fundamental de la herencia. Es una secuencia ordenada de nucleótidos localizados en una posición particular, sobre un cromosoma particular, que codifica un producto funcional específico (p.e. una proteína o una molécula de RNA).

Genoma

Todo el material genético en los cromosomas de un organismo particular, el tamaño de un genoma se da generalmente por el número total de pares de bases.

Genómica

La genómica es la rama de la biología que se encarga del estudio de los genomas. Se considera a un genoma como el conjunto de información genética (ADN) de un organismo.

Guanina

Una base nitrogenada, un miembro del par de bases G-C (guanina-citocina).

Homología

Similitudes en ADN o secuencias de proteínas entre individuos de la misma especie o entre diferentes especies. In vitro. Afuera de un organismo viviente.

Intrones

La secuencia de bases de ADN que interrumpen la secuencias de proteína codificante de un gen, estas secuencias se transcriben dentro del RNA, pero se cortan fuera del mensaje, antes de que se trasladen como proteínas, razón por la que también se les conoce como secuencias interpuestas.

Macromoléculas

Son moléculas que tienen una masa molecular elevada, formadas por un gran número de átomos. Generalmente podemos describirlas como la repetición de una o unas pocas unidades mínimas o monómeros, formando los polímeros. A menudo el término

macromolécula se refiere a las moléculas que contienen más de 100 átomos. Pueden ser tanto orgánicas como inorgánicas, y se encuentran algunas de gran relevancia en el campo de la bioquímica, al estudiar las biomoléculas. Dentro de las moléculas orgánicas sintéticas se encuentran los plásticos.

Material Genético

Ver *Genoma.*

Matriz

Una matriz es un conjunto de elementos de cualquier naturaleza aunque, en general, suelen ser números ordenados en filas y columnas.
Se llama matriz de orden "m × n" a un conjunto rectangular de elementos aij, dispuestos en filas "m" y en columnas "n". El orden de una matriz también se denomina dimensión o tamaño, siendo m y n números naturales.

Membrana Celular

La membrana que universalmente envuelve al citoplasma de las células.

Memoria Asociativa

Una memoria asociativa tiene por objetivo: recuperar de manera perfecta patrones, a partir de patrones de entrada, que quizá estén alterados con algún tipo de ruido.

Memoria Asociativa Bidireccional

Consta de dos capas de elementos que están completamente interconectados entre capas. Las unidades pueden o no tener conexiones de retroalimentación consigo mismas.
Al igual que en otras redes neuronales, en la arquitectura de la memoria asociativa existen pesos que se asocian a las conexiones entre elementos de un proceso. A diferencia de otras arquitecturas, estos pesos se determinan por anticipado, así es posible identificar a todos los vectores de entrenamiento.

Memoria Autoasociativa

Si en cada asociación sucede que el patrón de entrada es igual al de salida, la memoria es autoasociativa; en caso contrario, la memoria es heteroasociativa; esto significa que

una memoria autoasociativa puede considerarse como un caso particular de una memoria heteroasociativa

Una memoria es autoasociativa si se cumple que: $x^{\mu} = y^{\mu} \quad \forall \mu \in \{1,2,\ldots..,p\}$

Memoria Hopfield

Se considera a la memoria de Hopfield como una derivación de las bidireccionales, aunque existe la duda que sea ésta la forma en que se originó la memoria de Hopfield. Las versiones son la memoria discreta de Hopfield y la memoria continua de Hopfield, dependiendo de si las unidades de salida son una función discreta o continua de las entradas, respectivamente.

Memorias Heteroasociativas

Si en cada asociación sucede que el patrón de entrada es diferente al de salida, la memoria es heteroasociativa; en caso contrario, es autoasociativa.
Una memoria es heteroasociativa si se cumple lo siguiente: $\exists \mu \in \{1,2,\ldots..,p\}$ para el que $x^{\mu} \neq y^{\mu}$

Microarreglos

Un microarreglo de ADN (del inglés DNA microarrays) es una superficie sólida a la cual se unen una serie de fragmentos de ADN. Las superficies empleadas para fijar el ADN son muy variables y pueden ser vidrio, plástico e incluso chips de silicio. Los arreglos de ADN son utilizadas para averiguar la expresión de genes, monitorizandose los niveles de miles de ellos de forma simultanea.

Mitocondria

Las mitocondrias son los orgánulos que se encuentran en prácticamente todas las células eucariotas (también hay en células gaméticas), encargados de suministrar la mayor parte de la energía necesaria para la actividad celular; actúan por tanto, como centrales energéticas de la célula y sintetizan ATP por medio de la fosforilación oxidativa. La mitocondria presenta una membrana exterior permeable a iones, metabolitos y muchos polipéptidos.

Molécula

Una molécula es una partícula formada por un conjunto de átomos ligados por enlaces covalentes, metálicos, o iónicos de forma que permanecen unidos el tiempo suficiente como para completar un número considerable de vibraciones moleculares

Morfología Matemática

La palabra morfología significa forma y estructura de un objeto. Para imágenes binarias se definen operaciones morfológicas. y con estas se constituye una herramienta de extracción de componentes de imagen útiles en la representación y descripción de la forma de las regiones.

Las operaciones básicas de la morfología matemática son: dilatación se puede simplificar a decir que es agregar pixeles a un objeto, hacerlo mas grande, y luego erosión es hacerlo mas chico. La erosión saca los "outlayers del objeto".

Luego la combinación de estas dan origen a los operadores Apertura y Clausura. El primero consiste en aplicar una erosión seguida de una dilatación aplicando la misma forma estructurante, como resultado esta tiende a "abrir pequeños huecos". La clausura es el la aplicación de las operaciones básicas en el sentido inverso, y resulta en "cerrar los huecos".

Motifs

En genética, una secuencia de motifs es una secuencia de patrones de nucleótidos o aminoácidos, que es extensa y es, o se cree que es, biológicamente significativa. Para las proteínas, una secuencia de motifs se distingue por una estructura de motifs, un motif formado por un arreglo tridimensional de aminoácidos, que pueden no ser adyacentes.

Mutación

Cualquier cambio heredable en la secuencia de ADN. Comparar con polimorfismo.

Núcleo

En biología celular y citología: al núcleo celular, parte central de la célula rodeada de una membrana propia, llamada membrana nuclear, que contiene el ácido desoxirribonucleico (ADN o en inglés DNA) celular, donde se encuentran codificados los genes.

Nucleótidos

Molécula formada por una base nitrogenada (A, G, P, C) con azúcar de 5 carbonos y un grupo fosfato. El ácido nucleico es un polímero de muchos nucleótidos.

Pares Bases

Dos bases nitrogenadas (Adenina y Timina o Guanina y Citosina) que se mantienen unidas por medio de doble enlaces o puentes. Las dos cadenas de ADN se mantienen unidas en forma de doble hélice por enlaces entre los pares de bases.

Patrones

Son representaciones abstractas de un concepto en el mundo físico. Los cuales exhiben cierta regularidad en una colección de observaciones conectadas en el tiempo, en el espacio, o en ambas, y que pueden servir como modelo.

Péptidos

Los péptidos son un tipo de moléculas formadas por la unión de varios aminoácidos mediante enlaces peptídicos.

Perceptron

Un perceptrón se refiere a una neurona artificial y también como a la unidad básica de inferencia en forma de discriminador lineal, que suele formar parte de una red neuronal artificial.
Un perceptrón puede clasificar datos que sean linealmente separables. En el caso de un perceptrón con dos entradas deberá poder trazarse una única línea que separe las dos clases que permite identificar el perceptrón.

Polipéptidos

Un polipéptido es un péptido formado por una cadena simple de más de 10 aminoácidos y de menos de 50 aminoácidos. Varias cadenas de polipéptidos se pueden asociar para formar las proteínas.

Procariontes

Célula u organismo carente de una membrana definida, núcleo estructuralmente discreto así como de otros compartimentos celulares. Las bacterias son procariontes. Compárese con eucariontes.

Promotores

Un sitio sobre el ADN en el cual la RNA polimerasa se enlaza para iniciar la transcripción.

Proteínas

Una gran molécula compuesta de una o más cadenas de aminoácidos en un orden específico; el orden está determinado por la secuencia de bases de nucleótidos en el gen que codifica la proteína. Las proteínas se requieren para la estructura, función y regulación de las células corporales, tejidos y órganos y cada proteína posee funciones únicas. Ejemplos son las hormonas, enzimas y anticuerpos.

Proteómicas

La proteómica puede definirse como la genómica funcional a nivel de proteínas. Es la ciencia que correlaciona las proteínas con sus genes, estudia el conjunto completo de proteínas que se pueden obtener de un genoma. La proteómica intenta resolver preguntas como: ¿Qué función tienen las proteínas?, ¿Qué tipo de modificaciones postransduccionales sufren las proteínas y cuál es su función?, ¿Cómo varían las proteínas de una célula enfrentada a distintas condiciones ambientales?

Purina

Una base nitrogenada, contiene un solo anillo púrico y la contiene los ácidos nucleicos. La purina en el ADN es la adenina y en el RNA es la guanina.

Reconocimiento de Patrones

El reconocimiento de patrones, también llamado lectura de patrones, identificación de figuras y reconocimiento de formas1 es el reconocimiento de patrones en señales. No sólo es un campo de la informática sino un proceso fundamental que se encuentra en casi todos las acciones humanas.

El punto esencial del reconocimiento de patrones es la clasificación: se quiere clasificar una señal dependiendo de sus características. Señales, características y clases pueden ser de cualquiera forma.

Ribosa

Es el componente del ácido ribonucleico y otras sustancias como nucleótidos

Secuencia

Forma en que suceden o encadenan los nucleótidos a lo largo de las cadenas de ADN o RNA.

Similitud

Se presenta cuando dos objetos tienen atributos cuyos que les son comunes y cuyas medidas o cualidades relativas se ajustan a un grado de variación.

Sitios de Empalme

Las zonas de cambio entre un exon-intron (EI) o intron-exon (IE) son conocidos como sitios de empalme (splice-sites)

Técnicas Heurísticas

Una heurística es un algoritmo que ofrece uno o ambos objetivos; por ejemplo, normalmente encuentran buenas soluciones, aunque en ocasiones no hay pruebas de que la solución no pueda ser arbitrariamente errónea; o se ejecuta razonablemente rápido, aunque no existe tampoco prueba de que deba ser así.

Timina

Una base nitrogenada, un miembro del par de bases A-T (adenina-timina).

Transcripción

La síntesis de una copia de RNA a partir de una secuencia de ADN (un gen) el primer paso en la expresión del gen. Proceso por el que la enzima RNA polimerasa cataliza la síntesis de a partir de una de las cadenas (la cadena molde) de la doble hélice de ADN.

Upstream

La técnica consiste en la activación de un gen reportero (s) por la acción de un factor de transcripción que se enlaza a la secuencia regulatoria "UAS" (en inglés, Upstream activating sequence) localizada en el promotor mas arriba del sitio de inicio de la transcripción.

El principio de la técnica es el siguiente, el factor de transcripción es separado en dos fragmentos, uno que reconoce UAS y el otro que promueve la activación de la maquinaria de transcripción. Cada fragmento es introducido a las proteínas cuya interacción se desea analizar, usando técnicas de ingeniería genética. Si las proteínas forman un complejo entre sí, los dos fragmentos del factor de trancripción se encontrarán y el gen reportero será tanscrito.

Vector

Matemáticamente un vector puede ser un conjunto de elementos ordenados entre sí pero a diferencia de un conjunto normal como el de los números naturales, éste está ordenado.

Así, se llama vector de dimensión n a una tupla de n números reales (que se llaman componentes del vector). El conjunto de todos los vectores de dimensión n se representa como $\mathbb{R}^n$(formado mediante el producto cartesiano).

Ácido Ribonucleico(RNA)

Un compuesto químico que se encuentran en el núcleo y citoplasma de las células; desempeña una importante función en la síntesis de proteínas y otras actividades químicas de la células. La estructura del RNA es muy similar a la del ADN. Existen varias clases de moléculas RNA, como mensajero, de transferencia y ribosomal así como otros muy pequeños, pero cada uno tiene propósitos diferentes.

Apéndices

Apéndice A: Simbología

$\mathbf{M}$	Memoria asociativa max
m_{ij}	ij–ésima posición de la memoria M
Δm_{ij}	Incremento en m_{ij}
$\mathbf{W}$	Memoria asociativa min
$w_{\mathbf{ij}}$	ij–ésima posición de la memoria W
$\mathbf{x}$	vector columna que corresponde a un patrón de entrada
$x_{\mathbf{i}}$	i-ésima posición del patron X
$\mathbf{y}$	Vector columna que corresponde a un patrón de salida
y_{i}	i-ésima posición del patron Y
$(\mathbf{x}, \mathbf{y})$	Asociación de un patrón de entrada con uno de salida
(x^k, y^k)	Asociación de la *k*-ésima pareja de patrones
$\{(x^\mu, y^\mu) \mid \mu = 1, 2, ..., p\}$	conjunto fundamental
$\mathbf{A}$	Conjunto al que pertenecen los vectores **x** y **y.** $\mathbf{A} = \{0,1\}$
$\mathbf{B}$	Conjunto de 3 elementos $\mathbf{B} = \{0,1,2\}$
p	Número de parejas del conjunto fundamental
n	Dimensión de los patrones de entrada
m	Dimensión de los patrones de salida
$\forall$	Cuantificador universal
$\in$	Pertenencia de un elemento a un conjunto
$\exists$	Cuantificador existencial
$\times$	Producto cruz (entre conjuntos)
$\cdot$	Producto usual entre vectores o matrices
$\vee$	Operador máximo
$(\mathbf{x}^\mu)^t$	Transpuesto del vector $\mathbf{x}^\mu$
$\tilde{\mathbf{x}}$	Versión alterada del patrón fundamental **x**
∇	Producto máximo
Δ	Producto mínimo
$\wedge$	Operador mínimo
α	Operador Alfa, operador base de la fase de aprendizaje de las memorias asociativas alfa beta
β	Operador beta, operador base de la fase de aprendizaje de las memorias asociativas Alfa-Beta
$\alpha(x, y)$	Operación binaria α con argumentos *x* y *y*

$\beta(x, y)$	Operación binaria β con argumentos x y y
med	Operador media
∇_α	Operador α *max*
∇_β	Operador β *max*
Δ_α	Operador α *min*
Δ_β	Operador β *min*
$\boxtimes$	Símbolo que representa a las dos operaciones ∇_α y Δ_α
$\mathbf{V}$	Memorias asociativas Alfa-Beta tipo *max*
v_{ij}	ij–ésima posición de la memoria $\mathbf{V}$
$\mathbf{\Lambda}$	Memorias asociativas Alfa-Beta tipo *min*
λ_{ij}	ij–ésima posición de la memoria $\mathbf{\Lambda}$
$\mathbf{Z}^+$	Conjunto de los números enteros positivos
U_h	Suma de los componentes iguales a 1 de un vector x^h
s	Vector suma max
s_i	i-ésima posición del vector s
$\sim \mathbf{x}^\alpha$	Vector negado de $\mathbf{x}^\alpha$
C_h	Suma de los componentes iguales a 0 de $\mathbf{x}^h$
r	Vector suma min
r_i	i-ésima posición del vetor r
I	Vector Intermedio
I_i	i-ésima posición del vector intermedio
V^l	l-ésima memoria asociativa tipo max
Λ^l	l-ésima memoria asociativa tipo max
ρ	Operador de particionamiento vectorial
$\Diamond_A$	Operación mediana con operador A
$\Diamond_B$	Operación mediana con operador B

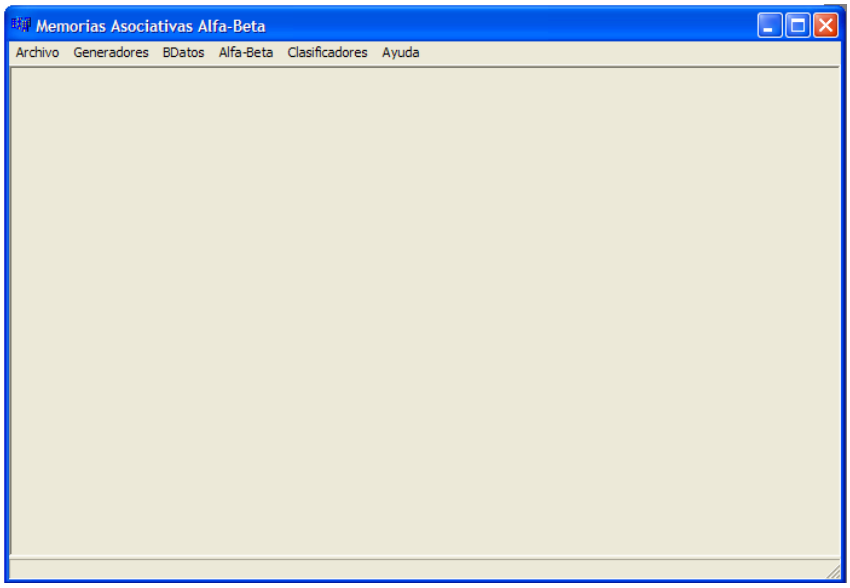
Apéndice B: Manual de usuario

En este apéndice se describe el funcionamiento del sistema MAB. El objetivo principal del sistema es brindar al usuario de las herramientas básicas necesarias para poder interactuar con el modelo de Memorias Asociativas Alfa-Beta.

Entre las herramientas que contiene son las siguientes

- Generador de patrones aleatorios
- Generador de clases
- Modelo de Memoria Heteroasociativa Alfa-Beta
- Modelo de Memoria Autoasociativa Alfa-Beta
- Nuevo algoritmo de Memoria Heteroasociativa Alfa-Beta
- Un Clasificador de Promotores

Al ejecutarse el programa se muestra el menú principal



Dicho menú contiene 5 opciones:

- Archivo:
- Generadores
- BDatos
- Alfa-Beta
- Clasificadores
- Ayuda

Cada opción a su vez contiene más opciones que nos permiten interactuar con el programa.

Archivo:

- Nuevo
- Abrir
- Guardar
- Guardar como
- Salir

Generadores:
- Patrones
- Clases

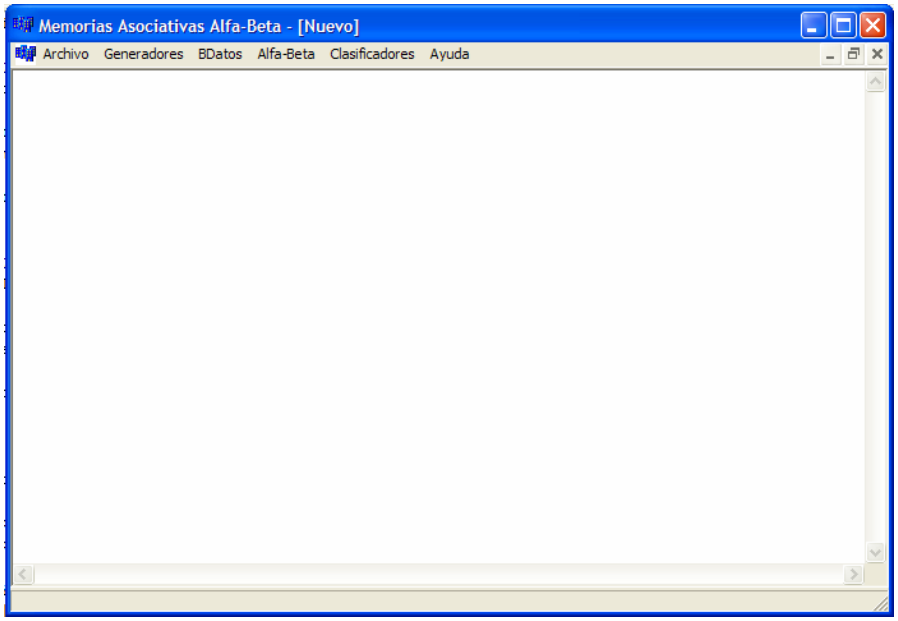
BDatos:
- Cargar formato
- Decimales
- Bioinformática

Alfa-Beta
- Heteroasociativas
- Autoasociativas

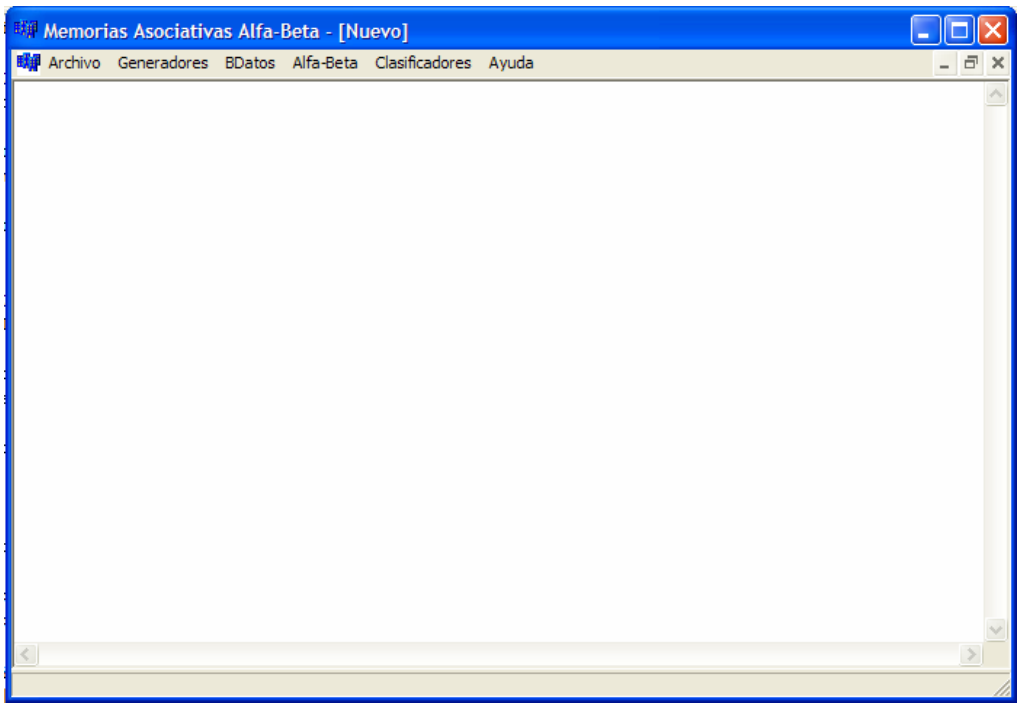
Clasificadores
- Iris-Plant
- Bioinformática

Ahora para poder trabajar e interactuar con los modelos Alfa-Beta se tiene que seguir los siguientes pasos:

1. Click en Archivo->Nuevo: esto desplegara un contenedor de texto sobre el cual se colocara toda la información que se genere en la fase de aprendizaje como de recuperación de la memoria.

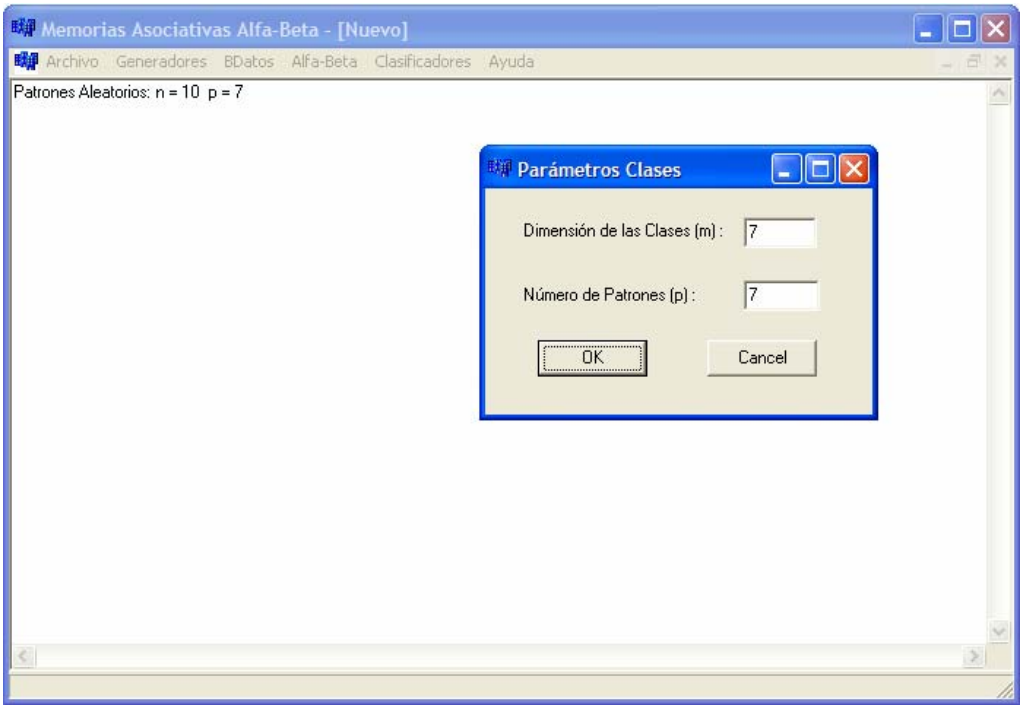


2. Una vez que se ha se ha abierto el contenedor las en Generadores se activan permitiendo generar un numero **p** de patrones de dimensión **n.**

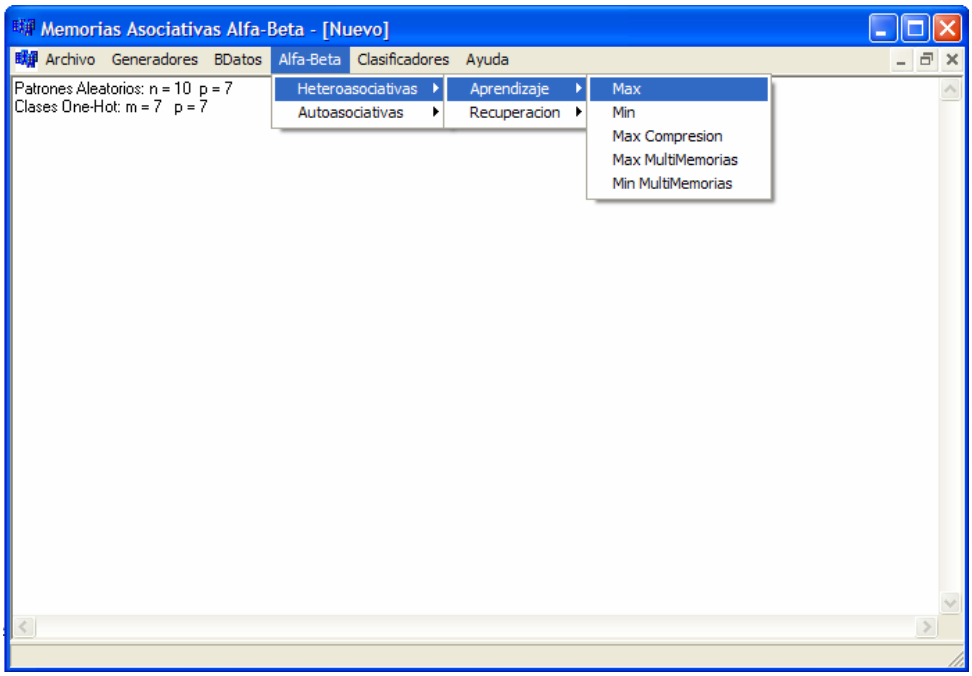


‘

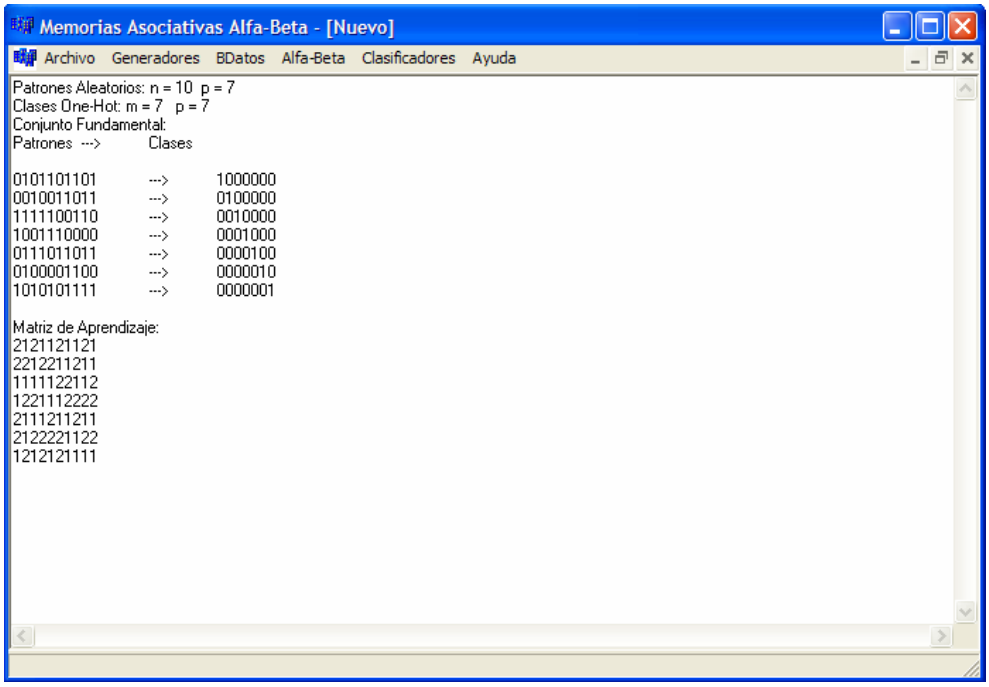
3. Una vez que se han creado los patrones es posible generar las clase, que pueden ser clases aleatorias del mismo tipo que los patrones aleatorios, o clases One-Hot o Cero-Hot.



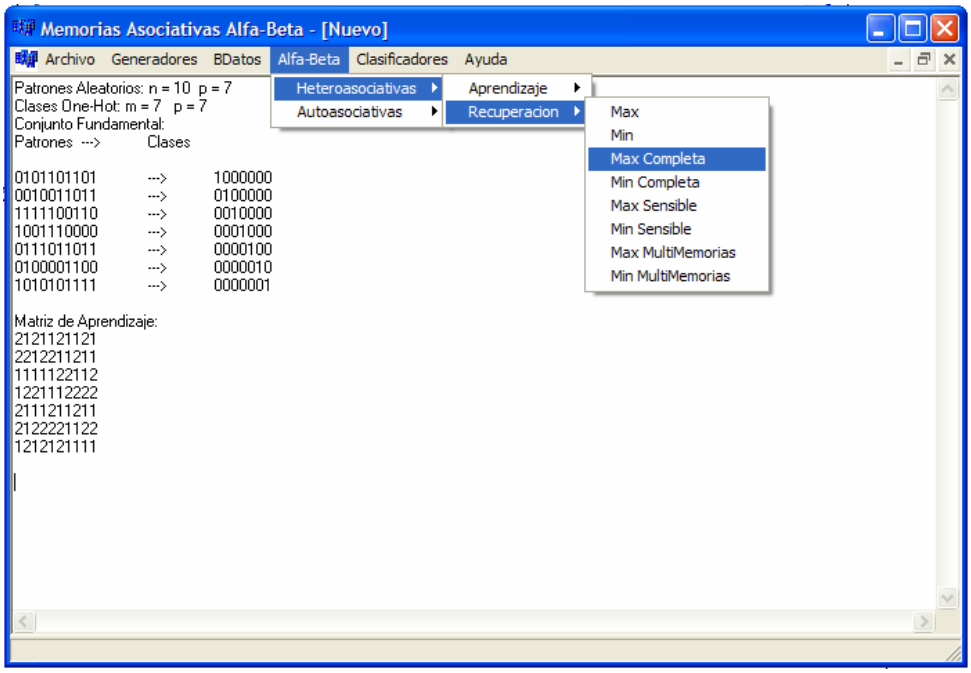
4. Una vez que se han generado las clases ya se tiene el conjunto fundamental listo para entrenar a la matriz ahora solo es cosa de seleccionar cual es el algoritmo de aprendizaje que requiera, en este caso seleccionaremos la memoria heteroasociativa Max.



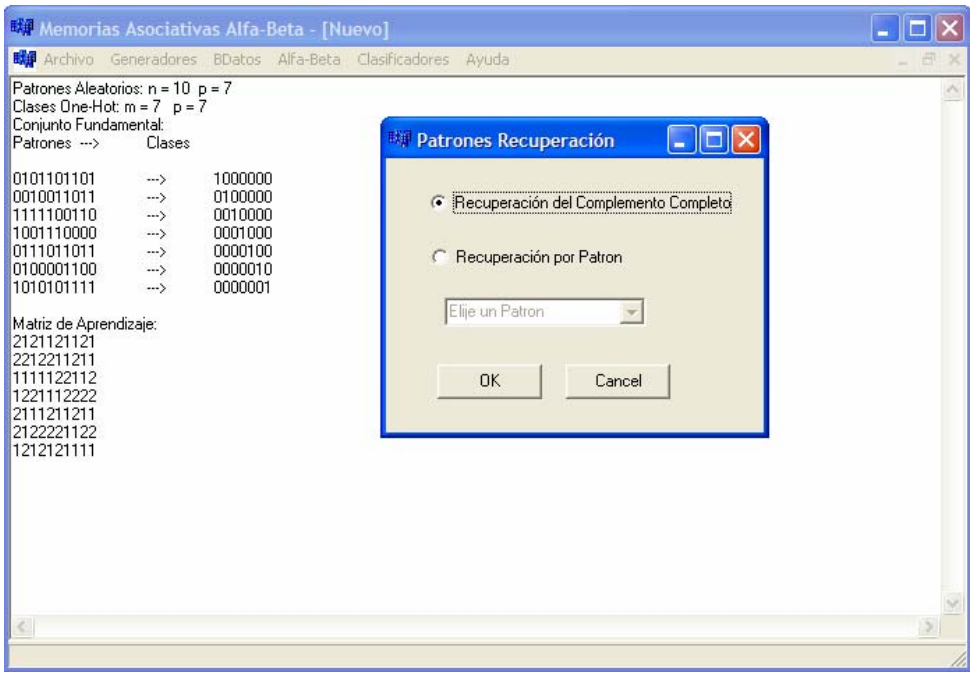
5. Una vez que se da aceptar se genera la matriz de aprendizaje mostrando el conjunto fundamental que se formó y la matriz de aprendizaje de dicho conjunto fundamental.

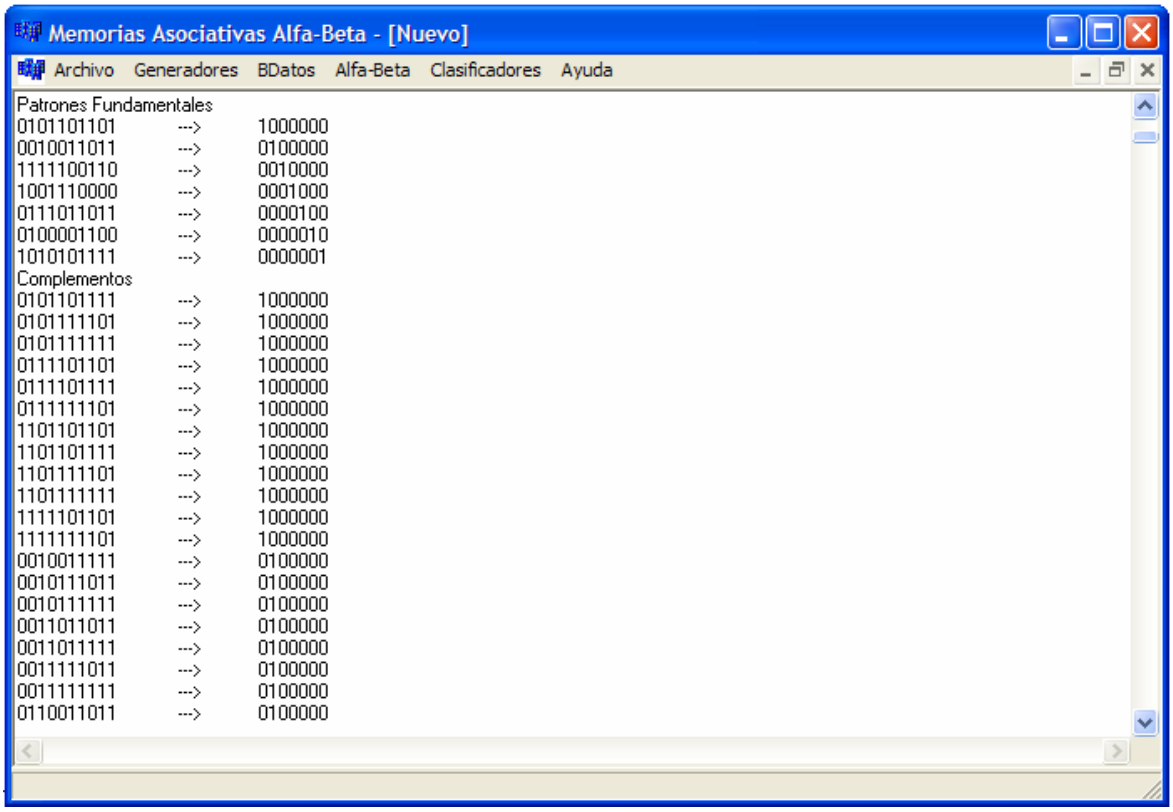


6. Ahora sólo queda seleccionar cual de los métodos de recuperación se va a aplicar a la matriz y de acuerdo al seleccionado será el porcentaje de aciertos en la recuperación. En este caso seleccionaremos la recuperación heteroasociativa perfecta que es una de las propuestas en esta tesis.



7. Y al aceptar se muestra en el contenedor de texto el complemento del conjunto fundamental el cual será el conjunto de entrenamiento.





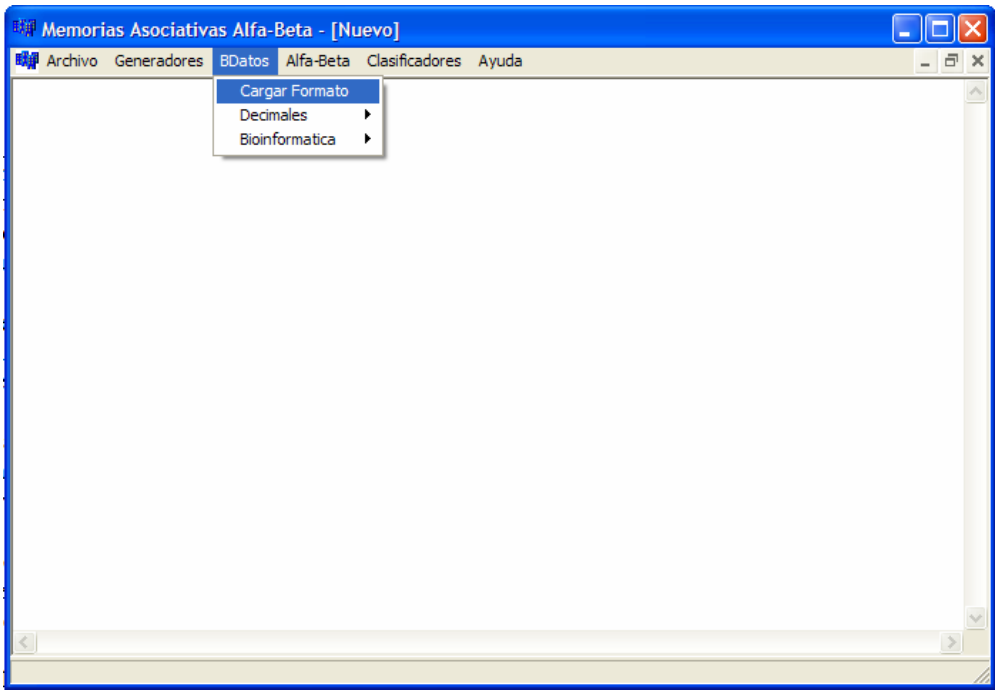
Como se puede ver, la última imagen muestra algunos de los patrones recuperado, los primeros que recupera de forma correcta son los del conjunto fundamental y el resto son patrones que no pertenecen el conjunto por lo tanto no habrá una contundencia en su recuperación.

El resto de los métodos llevan la misma secuencia de paso por lo que no se explicaran a detalle. Sin embargo, en el software existe otra posibilidad, la de cargar datos desde una base de datos.

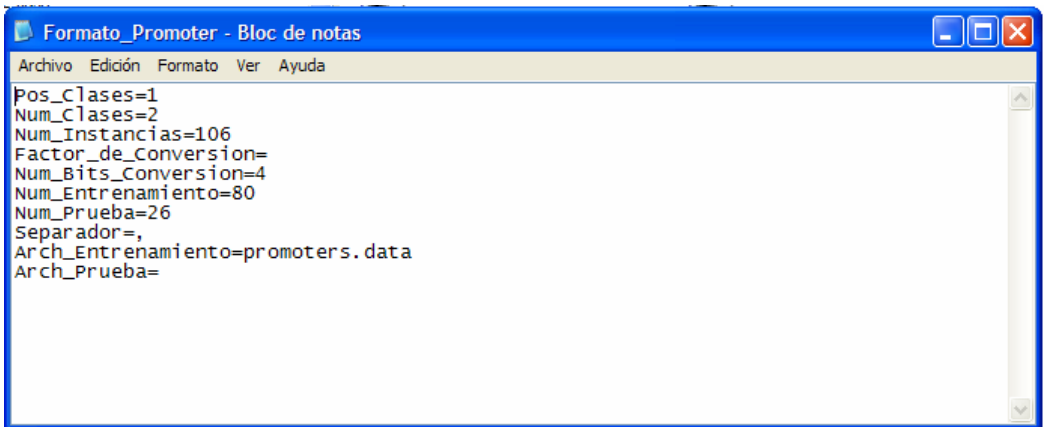
Dada la gran variedad de formatos en las bases de datos, en este sistema se implementan como ejemplo sólo la carga de dos bases de datos; la primera es la Iris-plant y la segunda es sobre promotores, y es sobre esta última en la que enfocaremos nuestra atención.

Dado que esta tesis ofrece un clasificador de patrones aplicado en la Bioinformática en este sistema se anexo el clasificador que muestra la efectividad del sistema. A continuación se presentan como es que se lleva cabo la carga, entrenamiento y prueba del sistema.

1. Primero se carga la base de datos accediendo de la siguiente forma:



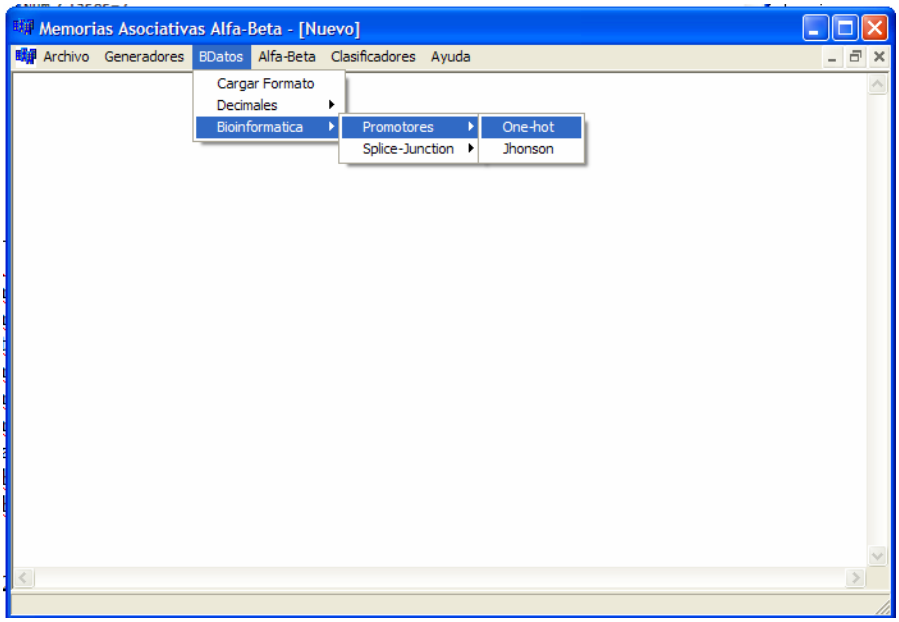
La información es cargada una vez que se selecciona un archivo con nombre Formato_Promoter, que contiene la información de la base de datos, y que es relevante para el sistema. Por lo que antes de cargarla es necesario que se llenen los campos a conveniencia del usuario.



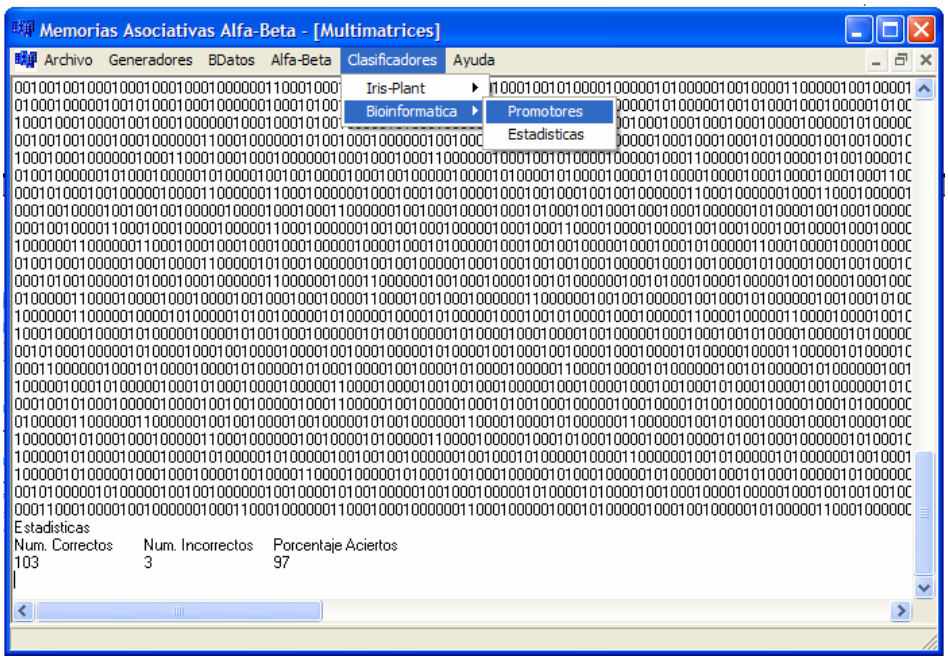
Pos_Clases: Posición de la clase en la base de datos
Num_Clases: Número de clases
Num_Instancias:Instancias que contiene la base de datos
Factor_de_Conversion: Factor para la conversión (en caso de ser decimal el atributo)
Num_Bits_Conversion=Numero de bits en los que se van a representar los datos.
Num_Entrenamiento=Tamaño del conjunto fundamental
Num_Prueba=Tamaño del conjunto prueba
Separador=Tipo de separador que se usa en la base de datos

Arch_Entrenamiento=Archivo de la base de datos
Arch_Prueba= Archivo de los patrones de entrenamiento (si existe por separado).

2. Una vez que se cargo la información solo resta crear el conjunto fundamental y para hacerlo sólo es necesario dar un click sobre el botón Promotores en el tipo de codificación que se desee para los patrones.



3. Ya que se tiene la codificación sólo resta dar un click en Clasificadores->Bioinformática->Promotores, y con esto se lleva a cabo el proceso de clasificación.



En esta última imagen se muestra la ubicación del botón “*promotores”* y la forma en la que se muestran los resultados. Al final del archivo muestra una estadística que contiene el número de patrones clasificados correctamente, incorrectamente y un porcentaje.

Debido a que los formatos en las bases de datos son muy variados, no es imposible crear un software que cumpla con las expectativas de cargar cualquier base de datos, es por eso que en este caso sólo se muestra la base de dato: la de promotores.