INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE INGENIERÍA MECÁNICA Y ELECTRICA

UNIDAD PROFESIONAL "ADOLFO LÓPEZ MATEOS"

"IMPLEMENTACIÓN DE UN AGENTE INTELIGENTE CON MEMORIAS ASOCIATIVAS"

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

INGENIERO EN COMUNICACIONES Y ELECTRÓNICA

PRESENTA:

MARIA DEL YANEC GUDIÑO MÉNDEZ

ASESORES:

M. EN C. MARÍA ELENA ACEVEDO MOSQUEDA

M. EN C. MARCO ANTONIO ACEVEDO MOSQUEDA

CIUDAD DE MÉXICO, FEBRERO 2016

INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE INGENIERÍA MECÁNICA Y ELÉCTRICA
UNIDAD PROFESIONAL "ADOLFO LÓPEZ MATEOS"

T E M A D E T E S I S

QUE PARA OBTENER EL TÍTULO DE	**INGENIERO EN COMUNICACIONES Y ELECTRÓNICA**
POR LA OPCIÓN DE TITULACIÓN	**TESIS Y EXÁMEN ORAL INDIVIDUAL**
DEBERA DESARROLLAR	**C. MARIA DEL YANEC GUDIÑO MENDEZ**

"IMPLEMENTACIÓN DE UN AGENTE INTELIGENTE CON MEMORIAS ASOCIATIVAS"

CREAR UN AGENTE INTELIGENTE QUE IMITE LAS ACCIONES QUE REALIZARÍA UNA PERSONA USANDO LAS MEMORIAS ASOCIATIVAS ALFA-BETA, LOGRANDO CON ELLO OBTENER UN CORRECTO APRENDIZAJE DADO LA EXCELENTE RECUPERACIÓN.

- **INTRODUCCIÓN AL PROYECTO A DESARROLLAR**
- **ESTADO DEL ARTE**
- **ANTECEDENTES TEÓRICOS**
- **DESARROLLO DEL PROYECTO**
- **PRUEBAS Y RESULTADOS**

CIUDAD DE MÉXICO., A 24 DE FEBRERO DE 2016.

A S E S O R E S

M. EN C. MARIA ELENA ACEVEDO MOSQUEDA | **M. EN C. MARCO ANTONIO ACEVEDO MOSQUEDA**

ING. BEATRIZ ADRIANA JAIME FONSECA

ING. PATRICIA LORENA RAMÍREZ RANGEL
JEFE DEL DEPARTAMENTO ACADÉMICO DE COMUNICACIONES Y ELECTRÓNICA

AGRADECIMIENTOS:

A MIS PADRES:

Por su amor y apoyo incondicional, por creer siempre en mí, por estar siempre a mí lado, sin importar la situación o la dificulta enfrentada, en verdad que sin ustedes no lo hubiera logrado. El camino fue largo y difícil, y aun cuando en momentos parecía no tener fin, se logró llegar a la meta, pero no al final del camino, ya que esto es solo el principio de una nueva y emocionante etapa, no sé qué me depare el futuro, pero con su apoyo todo es posible, gracias por ser mis padres.

A MÍ HERMANO:

Por su cuidado, amor y Apoyo, eres mi hermanito querido y adorado, gracias por estar y gracias por existir.

A MÍS PROFESORES:

Por la dedicación, consejos y ánimos que siempre dieron.

A DÍOS:

Porque aun no siendo tan cercana a la iglesia y a sus creencias, reconozco que hay una fuerza superior, que siempre está con nosotros en cada momento, gracias por darme una familia tan única y especial.

ING. MARIA DEL YANEC GUDIÑO MÉNDEZ.

RESUMEN

A través del presente trabajo se pretende demostrar la utilidad de las memorias asociativas BAM Alfa y Beta para la impletentación de un agente inteligente.

El presente proyecto tiene como inspiración el proyecto de "A computational Model of Imitation and Autonomous Behavior in Continuous Spaces" propuesto por Tasuya Sakato, Motoyuki Ozeki y Natsuki Oka en el año 2013. En el a través del método de aprendizaje por refuerzo un agente inteligente logra imitar los movimientos y acciones de un agente optimo, del cual se toman las decisiones correctas respecto a que acción ejecutar dependiendo el objeto que se identifica en el entorno simulado. Los objetos a identificar son una manzana, un racimo de plátanos y una piedra. Las acciones a ejecutar dependiendo el tipo de objeto son comer y arrojar.

A través de este proyecto se muestra como el agente inteligente logra reconocer una mayor de objetos, tanto comestibles como no comestibles, por lo tanto el rango de acciones a ejecutar aumentara. Esto se lograra plasmando las características de cada objeto en bits, lo mismo ocurre con las acciones a ejecutar, de modo que se tenga un patrón de entrada $\boldsymbol{x}$ y un patrón de salida $\boldsymbol{y}$. Al lograr esto, se utilizaran las memorias Asociativas BAM Alfa-Beta para conseguir una correcta recuperación de patrones.

Es importante mencionar que este proyecto solo se lleva a cabo a nivel software, apoyándonos en el software de Matlab para realizar el programa correspondiente y ejecutar las acciones.

Se espera poder implementarse posteriormente a nivel hardware con un agente inteligente que pudiese realizar dichas acciones físicamente.

CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

SOY POLITÉCNICO

Porque aspiro a ser todo un hombre

CAPÍTULO 1

INTRODUCCIÓN

1.1 Contexto

La Inteligencia Artificial (IA), es hoy por hoy una rama de las ciencias de la computación que más progreso ha aportado. A su vez, las memorias asociativas han demostrado tener un alto índice de recuperación de patrones logrando con ello un aprendizaje preciso. Es por esto que la aplicación de memorias asociativas en la creación de un agente inteligente por imitación nos parece ser de gran utilidad tomando en cuenta la cantidad de aprendizaje que logran retener optimizando costos y tiempo de trabajo.

1.2 Planteamiento del Problema

Actualmente, el diseño de un Agente Inteligente ha sido orientado al uso de diversos métodos de aprendizaje, el más común es a través de prueba y error, es decir aprendizaje por refuerzo, el cual requiere mayor tiempo y por lo tanto limita el número de acciones a realizar.

Recientemente se ha recurrido al uso del aprendizaje por imitación combinado con el aprendizaje por refuerzo, por ejemplo, en un entorno donde se encuentran dos agentes (en lo sucesivo agente óptimo y agente de aprendizaje),

si el agente óptimo realiza alguna acción, el agente de aprendizaje observara esa la acción y elegirá entre el módulo de aprendizaje por refuerzo, o el módulo de imitación. Para lograr esto, primero se tienen que crear estados y acciones, se seleccionaran los objetos a imitar, se crearán políticas de imitación, se seleccionarán las acciones a ejecutar y por último se deberán actualizar las acciones y políticas a realizar.

Todo este proceso consume mucho tiempo y de alguna manera limita el número de acciones que se pueden realizar.

1.3 Justificación

Se tomaron como base las memorias asociativas Alfa-Beta bidireccionales (BAM); puesto que han demostrado obtener una recuperación perfecta de patrones, obteniendo con ello un aprendizaje total que es excelente para el objetivo de nuestro trabajo. Debido a ello, la imitación de las acciones a realizar del agente inteligente siempre son las correctas y el proceso no involucra tantos eventos

1.4 Hipótesis

Por medio del uso de las BAM Alfa y Beta se puede obtener una completa recuperación de patrones logrando con ello obtener una correcta imitación y un completo aprendizaje de las acciones a plantear.

1.5 Objetivo.

Crear un Agente Inteligente que imite las acciones que realizaría una persona usando las BAM Alfa-Beta, logrando con ello obtener un correcto aprendizaje dado la excelente recuperación.

1.6 Alcance del Proyecto

Con la implementación de las BAM Alfa-Beta en la creación de un agente inteligente, se podrían abrir nuevas formas de demostrar como el uso de las memorias asociativas Alfa-Beta bidireccionales pueden completar la aportación de este trabajo, y demostrar la alta eficiencia que proporcionan las memorias asociativas de tipo BAM, en la creación de un agente inteligente con el uso de ellas.

A su vez, demostrar que si todas las características de un objeto son plasmadas en bits, para el procesamiento de este en una memoria asociativa, al introducirlo como patrón de entrada, se obtendrá una correcta asociación con su correspondiente salida.

Con esto, el número de objetos que se pueden procesar aumenta, y a su vez también las acciones que se pueden ejecutar son mayores, aumentando la funcionalidad del agente inteligente.

1.7 Estructura del proyecto

En el capítulo 1 se da a conocer el por qué se planteó este problema, así como cuáles son los alcances y contribuciones que este puede aportar a la rama de la Inteligencia Artificial(IA), haciendo uso de las BAM alfa y beta.

En el capítulo 2 se da una introducción al agente inteligente, así como los diversos tipos de entornos en los que se puede situar. Además de mencionar los diversos tipos de métodos de aprendizaje que se han desarrollado y sus aplicaciones.

En el capítulo 3 se hace un repaso a los antecedentes de las memorias asociativas, desde las clásicas, hasta llegar a las BAM Alfa-Beta.

El capítulo 4 se describe a profundidad nuestra propuesta de diseño, así como el desarrollo de nuestro proyecto.

Y por último en el capítulo 5 se plasman los resultados obtenidos con nuestra propuesta.

SOY POLITÉCNICO

Porque exijo mis deberes antes de mis derechos

CAPÍTULO 2

ESTADO DEL ARTE

2.1 ¿Qué es la IA?

Hoy por hoy la inteligencia artificial (IA), es considerada un área multidisciplinaria, que haciendo uso de ciencias como la lógica, la filosofía y la informática, estudia la creación y diseño de agentes no vivos que imiten la inteligencia humana.

John McCarthy [1] acuño el término inteligencia artificial por primera vez en 1956 durante la conferencia *"The Dartmouth Summer Research Project on Artificial Intelligence"*, en la cual la definió de la siguiente manera:" La Inteligencia Artificial es la ciencia e ingenio de hacer máquinas inteligentes, especialmente programas de cómputo inteligentes".

Dentro de la IA podemos encontrar diversos campos de investigación, tales como la ingeniería de software, el procesamiento de lenguaje natural, la robótica, redes neuronales y memorias asociativas.

2.2 Agente Inteligente

Un agente es algo que razona (agente viene del latín *agere*, que significa hacer). Un agente inteligente puede ser una entidad física o virtual y se define como aquel sistema capaz de percibir su entorno y actuar sobre él, por ejemplo en la rama de la

computación es un sistema computacional capaz de actuar de manera autónoma para satisfacer sus objetivos. La eficiencia de un agente Inteligente se mide sobre los resultados en un determinado entorno, calificando su capacidad de recuperación y elección correcta.

Por lo que podemos concluir que un agente inteligente es una entidad software que, basándose en su propio conocimiento, realiza un conjunto de operaciones para satisfacer las necesidades de un usuario o de otros programas.

2.2.2 Tipos de entorno

El rango de los entornos de trabajo en los que se utilizan técnicas de IA es muy grande. Sin embargo, se puede identificar un pequeño número de dimensiones para categorizar estos entornos. Estas dimensiones determinan, según Stuart J. Russell y Peter Norving [2], hasta cierto punto, el diseño más adecuado para el agente y la utilización de cada una de las familias principales de técnicas en la implementación de este:

A continuación se nombran estas dimensiones:

- Totalmente observable contra parcialmente observable.
  Un entorno es totalmente observable siempre y cuando los sensores con que cuenta el agente sean capaces de detectar todos y cada uno de los aspectos que son relevantes en la toma de decisiones. Mientras que se considera parcialmente observable debido al ruido, y la existencia de sensores poco adecuados para la detección correcta de elementos.

- Determinista contra estocástico.
  Si el siguiente estado del medio está totalmente determinado por el estado actual y la acción ejecutada por el agente, entonces se dice que el entorno es determinista, de otra forma es estocástico.

- Episódico contra secuencial.
  En un entorno de trabajo episódico, cada acción corresponde a el episodio que se está viviendo, y ninguno de estos afectara a los siguientes episodios en sí.

Mientras que en un escenario secuencial, la decisión tomada en el presente podría traer afectar a las decisiones futuras.

- Estático contra dinámico
  Si el entorno puede cambiar mientras que el agente está tomando alguna decisión para actuar se dice que el entorno es dinámico, en caso contrario es estático. Si el entorno no cambia con el paso del tiempo, pero el rendimiento del agente si, se dice que el medio es semidinámico.

- Discreto contra continúo.
  Un entorno con estados discretos es aquel que posee un número finito de estados distintos. Un entorno continuo, posee un número infinito de estados continuos.

- Agente individual contra multiagente.
  Un agente individual es aquel que se encuentra solo en un entorno, es decir la toma de elecciones correctas depende solo de las circunstancias que se le presenten sin que intervenga algún ente. Mientras que se habla de multiagente cuando la toma de decisiones depende directamente de las acciones efectuadas por otro agente, en este caso se tendrá que evaluar la decisión correcta según la acción observada. Un ejemplo de esto es un agente jugando ajedrez, mientras que el agente individual se puede ejemplificar con uno resolviendo un crucigrama.

2.3 Métodos de aprendizaje y sus aplicaciones

La investigación sobre los diversos métodos de aprendizaje y los entornos en los que se desarrollan no ha parado. Por lo que aquí se muestran algunos ejemplos.

2.3.1 "Reinforcement Learning: An Introduction"

Sutton y Barto en 1998 en su *"Reinforcement Learning: An Introduction" [3]* exploran un enfoque computacional para el aprendizaje por medio de la interacción. En lugar de teorizar directamente acerca de cómo las personas o los animales aprenden, se analizan situaciones de aprendizaje idealizadas. Es decir, se exploran diseños para las máquinas que son eficaces en la solución de problemas de aprendizaje de interés científico o económico, la evaluación de estos diseños es a través del análisis matemático o experimentos computacionales.

Este tipo de enfoque es el llamado aprendizaje por refuerzo, el cual es mucho más centrado en la meta.

2.3.2 "Principled methods for advising reinforcement learning agents"

Una cuestión importante en el aprendizaje por refuerzo es la forma de incorporar el conocimiento experto en forma de principios, especialmente a medida que escalamos a las tareas del mundo real.

En este trabajo Wiewiora, Cottrell y Elkan" [4], nos presentan un método para incorporar consejos arbitrarios en la estructura de recompensa para un agente utilizando el método de aprendizaje por refuerzo sin alterar la política optima de este. Esto permite mucha más información específica para guiar al agente.

2.3.3 "Discovering optimal imitation strategies. Robotics and Autonomous Systems"

Se desarrolla una política general para el aprendizaje, haciendo uso de las características relevantes de la imitación.

Los autores Billard, Epars, Calinon, Schaal, y Cheng [5], estudian la imitación de las tareas a través de la manipulación o gestos. El proceso de imitación se modela como un sistema de optimización jerárquica, lo que minimiza la discrepancia entre los dos conjuntos de datos multidimensionales. Para clasificar a través de estrategias de manipulación, ellos aplican un análisis probabilístico de los datos en espacios cartesianos y conjuntos. Determinando una métrica general, que optimiza la política

de la reproducción de tareas, después de haber determinado la estrategia. El modelo descubre con éxito estrategias en seis tareas realizada por medio de imitación y la reproducción de una tarea diferente por un robot humanoide de cuerpo completo.

2.3.4" *Imitation with Alice: Learning to imitate corresponding actions across dissimilar embodiments*"

La imitación es un poderoso mecanismo por el cual el conocimiento puede ser transferido entre agentes (tanto biológicos como artificiales). Los principales problemas en el tema de la imitación han surgido zonas cercanas a la inteligencia artificial, incluyendo el cognitivo y las ciencias sociales, el comportamiento animal, la robótica, la interacción humano-computadora, ingeniería de software, la programación con el ejemplo y el aprendizaje automático. Sistemas artificiales que se utilizan para estudiar la imitación en ambos modelos de prueba, se derivan de los datos observacionales o neurobiológicos sobre la imitación de animales para luego poder aplicarla a diferentes tipos de sistemas no biológicos que van desde robots, a los agentes de software. Un problema crucial en la imitación es el de correspondencia, es decir, las secuencias de acción de mapeo del manifestante y del agente imitador. En este trabajo se describe un nuevo mecanismo general de imitación llamado ALICE [6] (aprendizaje en la acción para la imitación a través de la correspondencia entre las realizaciones) que trata específicamente el problema de la correspondencia. El mecanismo es aplicado y su eficacia se ilustra en el banco de pruebas " *ChessWorld* " que se creó para estudiar la imitación desde una perspectiva basada en agentes, es decir, para un agente en particular, en un entorno particular.

2.3.5" A computational Model of Imitation and Autonomous Behavior in Continuous Spaces"

Este modelo propuesto por Tasuya Sakato, Motoyuki Ozeki y Natsuki Oka en el año 2013 [7], supone un entorno en donde se encuentran dos agentes, uno óptimo y otro de aprendizaje. Se hace uso del aprendizaje por refuerzo (RL), por imitación, y el módulo de selección de acción.

El modelo propuesto es implementado para el agente de aprendizaje.

Cuando el agente óptimo realiza acciones, el agente de aprendizaje observa las acciones, los estados y recompensas que se obtienen por estas. El agente de aprendizaje determina una acción a realizar a partir de lo observado.

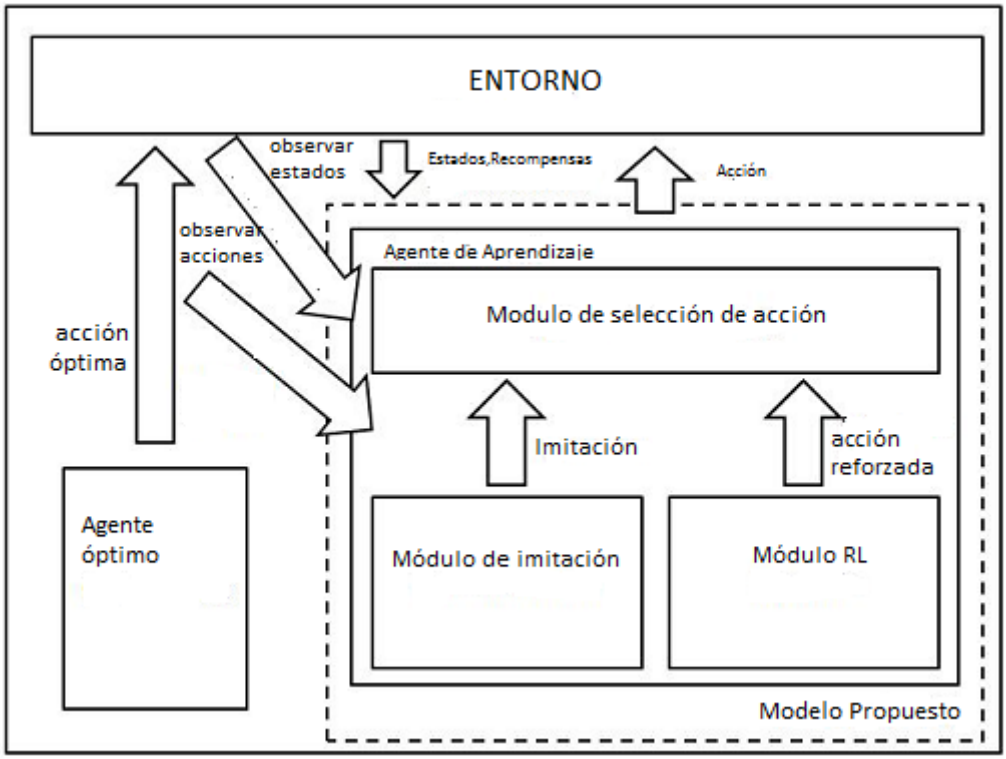


Figura 2.1 Configuración del Modelo propuesto en *A computational Model of Imitation and Autonomous Behavior in Continuous Spaces* [7].

El aprendizaje y el proceso de generación de la acción del agente de aprendizaje se describen a continuación:

1. Si el agente óptimo realiza alguna acción.
2. El agente de aprendizaje observa la acción y estado.
3. Se selecciona una acción de destino de las acciones observadas.
4. Elije entre dos alternativas: acción reforzada o imitación.
5. Si el agente eligió la imitación
6. Se selecciona una política de imitación de imitación.

7. Imita el objetivo de la política seleccionada.
8. Si no:
9. Realiza la acción reforzada.
10. Termina si:
11. Consigue recompensas.
12. Realiza aprendizaje por refuerzo
13. Actualiza los valores
14. Regresa al punto 1.

El aprendizaje por refuerzo se utiliza con el fin de aprender conducta autónoma, mientras que los estados y acciones observados se registran en el módulo de imitación.

Los experimentos se llevan a cabo en el siguiente entorno. Se utiliza un simulador de mesa de comedor, en el hay dos cabezas, así como sus manos derechas, una mesa y un objeto en el entorno.

Los tipos de objetos son una manzana, un racimo de plátanos y una piedra. Uno de los objetos es colocado al azar en la mesa como estado inicial. El agente de la parte inferior es el agente óptimo, y el de la parte superior es el agente de aprendizaje. Los agentes actúan cada episodio alternativamente, este termina si el objeto en el medio ambiente se come o es tirado. El agente de aprendizaje observa acciones y estados cuando el agente óptimo actúa.

En la figura 2.2 se ilustra el entorno en el que se desenvuelven las acciones.

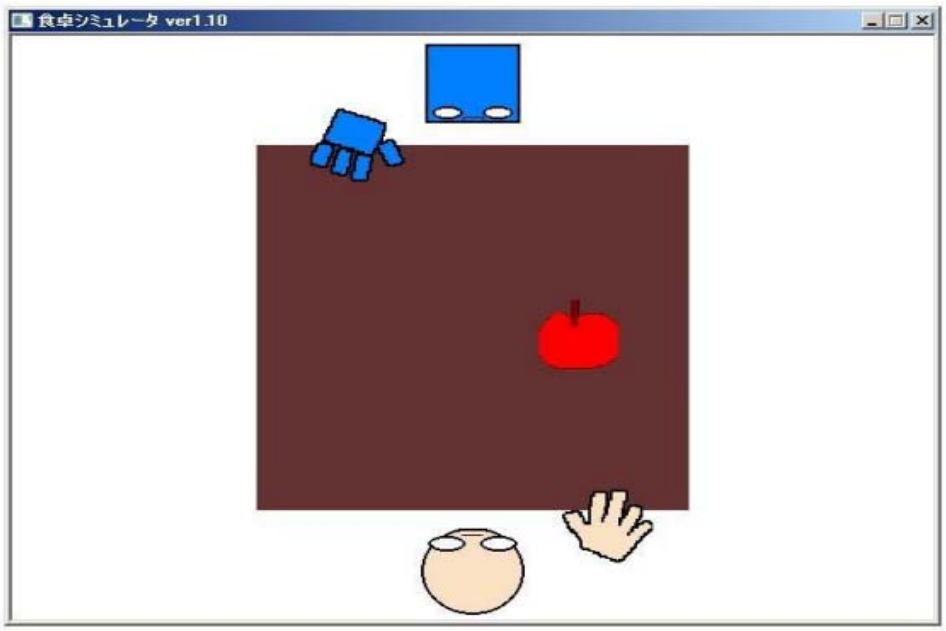


Figura 2.2. Simulación de entorno [7].

Cabe resaltar que este artículo es la inspiración principal de este proyecto.

Nuestra propuesta es incrementar el número de objetos a identificar por nuestro agente, y por consiguiente las acciones que este puede ejecutar, disminuyendo con el uso de las BAM Alfa-Beta el tiempo de aprendizaje.

En el capítulo 4 se retomara este artículo, para posteriormente dar inicio con la implementación de nuestro agente inteligente.

SOY POLITÉCNICO

Por convicción y no por circunstancia

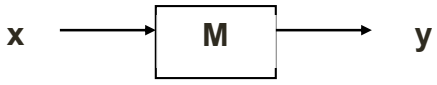
CAPÍTULO 3

ANTECEDENTES

3.1 Conceptos Básicos

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.
Los conceptos utilizados en esta sección se encuentran en la referencia [8].

Una **Memoria Asociativa** puede formularse como un sistema de entrada y salida, idea que se esquematiza a continuación:



En este esquema, los patrones de entrada y salida están representados por vectores columna denotados por **x** y **y**, respectivamente. Cada uno de los patrones de entrada forma una asociación con el correspondiente patrón de salida, la cual es similar a la una pareja ordenada; por ejemplo, los patrones **x** y **y** del esquema anterior forman la asociación (**x**,**y**).

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: este es el **conjunto fundamental de aprendizaje**, o simplemente **conjunto fundamental**.

El conjunto fundamental se representa de la siguiente manera:

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

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 $\mathbf{x}^{\mu} = \mathbf{y}^{\mu} \ \forall \mu \in \{1, 2, \ldots, p\}$, por lo que uno de los requisitos que se debe de cumplir es que $n = m$.

Una memoria **Heteroasociativa** es aquella en donde $\exists \mu \in \{1, 2, \ldots, p\}$ para el que se cumple que $\mathbf{x}^{\mu} \neq \mathbf{y}^{\mu}$. Nótese que puede haber memorias heteroasociativas con $n = m$.

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 asociativa opera sobre un patrón de entrada, a la manera del esquema que aparece al inicio de esta sección.

A fin de especificar las componentes de los patrones, se requiere 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.

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:

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

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

Una vez estudiados los conceptos básicos, es necesario analizar las dos fases que experimenta una memoria asociativa.

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 $\{(\mathbf{x}^1,\mathbf{y}^1), (\mathbf{x}^2,\mathbf{y}^2), \ldots, (\mathbf{x}^p,\mathbf{y}^p)\}$, donde $\mathbf{x}^{\mu} \in A^n$ y $\mathbf{y}^{\mu} \in A^m \; \forall \mu \in \{1, 2, \ldots, p\}$. Si $\exists \mu \in \{1, 2, \ldots, p\}$ tal que $\mathbf{x}^{\mu} \neq \mathbf{y}^{\mu}$, la memoria será heteroasociativa; si $m = n$ y $\mathbf{x}^{\mu} = \mathbf{y}^{\mu} \; \forall \mu \in \{1, 2, \ldots, p\}$, 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 $\omega \in \{1, 2, \ldots, p\}$, ésta responde con el correspondiente patrón fundamental de salida $\mathbf{y}^{\omega}$.

Una memoria asociativa bidireccional también es un sistema de entrada y salida, solamente que el proceso es bidireccional. La dirección hacia adelante se describe de la misma forma que una memoria asociativa común: al presentarle una entrada **x**, el sistema entrega una salida **y**. La dirección hacia atrás se lleva a cabo presentándole al sistema una entrada **y** para recibir una salida **x**.

3.2 Memorias Asociativas

Con la finalidad de establecer el marco de referencia en el que surgieron la memorias asociativas bidireccionales, se analizaran primeo los modelos de memoria asociativas.

Las memorias asociativas presentadas a continuación, son los modelos más representativos que sirvieron de base para la creación de modelos matemáticos que sustentan el diseño y operación de memorias asociativas más complejas. Para cada modelo se describe su fase de aprendizaje y su fase de recuperación.

Se incluyen cuatro modelos clásicos basados en el anillo de los números racionales con las operaciones de multiplicación y adición: *Lernmatrix*, *Correlograph*, *Linear Associator* y Memoria Hopfield, además de tres modelos basados en paradigmas diferentes a la suma de productos, a saber: memorias asociativas Morfológicas, memorias asociativas Alfa-Beta y memorias asociativas Media.

3.2.1 *Lernmatrix* de Steinbuch

Karl Steinbuch fue uno de los primeros investigadores en desarrollar un método para codificar información en arreglos cuadriculados conocidos como *crossbar* [9]. La importancia de la *Lernmatrix* [10,11] se evidencia en una afirmación que hace Kohonen [12] 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* [22]. El método funciona así: para representar la clase $k \in \{1, 2, \ldots, p\}$, 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 $j = 1, 2, \ldots, k\text{-}1, k\text{+}1, \ldots, p$.

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 $\mathbf{M}$, la *Lernmatrix* de Steinbuch, tiene valor cero al inicio, y se actualiza de acuerdo con la regla $m_{ij} + \Delta m_{ij}$, donde:

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

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*:

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

3.2.2 *Correlograph* de Willshaw, Buneman y Longuet-Higgins

El *correlograph* es un dispositivo óptico elemental capaz de funcionar como una memoria asociativa [13]. 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 $(\mathbf{x}^{\mu}, \mathbf{y}^{\mu}) \in A^n \times A^m$. 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:

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

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:

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

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: $log_2 n$

3.2.3 *Linear Associator* de Anderson-Kohonen

El *Linear Associator* tiene su origen en los trabajos pioneros de 1972 publicados por Anderson y Kohonen [12, 14].

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

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

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

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

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

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

Fase de Recuperación

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

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

3.2.4 La memoria asociativa Hopfield

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 $\{(\mathbf{x}^\mu, \mathbf{x}^\mu) \mid \mu = 1, 2, \ldots, p\}$ 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 $\mathbf{M}$:

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

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, $x_i(0) = \tilde{x}_i$, $\forall i \in \{1,2,3,\ldots,n\}$

2) $\forall i \in \{1,2,3,\ldots,n\}$ se calcula $x_i(t+1)$ de acuerdo con la condición siguiente:

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

3) Se compara $x_i(t+1)$ con $x_i(t)$ $\forall i \in \{1, 2, 3,\ldots,n\}$. 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 $x_i(\tau+1) = x_i(\tau)\ \forall i \in \{1, 2, 3,\ldots,n\}$; el proceso termina y el patrón recuperado es $\mathbf{x}(\tau)$.

3.2.5 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 [15].

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 $[c_{ij}]_{m\text{x}n}$ cuya *ij*-ésima componente c_{ij} es

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

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

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

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.

3.2.5.1 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 $\mathbf{M}$.

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

Fase de Recuperación

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

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

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

3.2.5.2 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:

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

3.2.6 Memorias Asociativas Alfa-Beta

Las memorias Alfa-Beta [16] 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:

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

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

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

La operación binaria β: $B \times A \rightarrow A$ 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

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.

3.2.7 Memorias Asociativas Media

Las Memorias Asociativas Media [17] utilizan los operadores A y B, definidos de la siguiente forma:

$A(x, y) = x - y$

$B(x, y) = x + y$

Las operaciones utilizadas se describen a continuación.

Sean $P = [p_{ij}]_{m\times r}$ y $Q=[q_{ij}]_{r\times n}$ dos matrices.

Operación $\Diamond_A$: $P_{m\times r} \Diamond_A Q_{r\times n} = [f_{ij}^A]_{m\times n}$ donde $f_{ij}^{A} = \underset{k=1}{\overset{r}{\mathbf{med}}} A(p_{ik}, q_{k,j})$

Operación $\Diamond_B$: $P_{m\times r} \Diamond_B Q_{r\times n} = [f_{ij}^B]_{m\times n}$ donde $f_{ij}^{B} = \underset{k=1}{\overset{r}{\mathbf{med}}} B(p_{ik}, q_{k,j})$

Algoritmo Memorias Media

Fase de Aprendizaje

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

$[\mathbf{y}^{\xi} \Diamond_A (\mathbf{x}^{\xi})^t]_{m\times n}$

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

$$\mathbf{M} = \underset{\xi=1}{\overset{p}{\mathbf{med}}} \left[y^{\xi} \lozenge_{\mathrm{A}} \left(x^{\xi} \right)^{t} \right]$$

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

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

Fase de Recuperación

Se tienen dos casos:

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

$$\mathbf{M} \lozenge_{\mathrm{B}} \mathbf{x}^{\omega}$$

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

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

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

$$\mathbf{M} \lozenge_{\mathrm{B}} \tilde{\mathbf{x}}$$

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

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

3.3 BAM Alfa-Beta

Las memorias asociativas bidireccionales Alfa-Beta (BAM Alfa-Beta) [8] han demostrado ser una de las contribuciones más importantes en esta área, ya que las características de los patrones entrenados, como: distancia de Hamming, ortogonalidad, independencia lineal, entre otras, no limitan el buen funcionamiento de la BAM Alfa-Beta [8].

Este modelo fue desarrollado por la Dra. María Elena Acevedo Mosqueda, asesora de esta tesis y son la base de esta tesis.

.

A continuación se explica su funcionamiento, resaltando que los conceptos básicos de las BAM Alfa-Beta se encuentran en la referencia no. [8]:

3.3.1 Descripción de las Memorias Asociativas Bidireccionales Alfa-Beta

En general, cualquier modelo de memoria asociativa bidireccional presente en la literatura científica actual se podría esquematizar como se muestra en la figura 3.3.1.

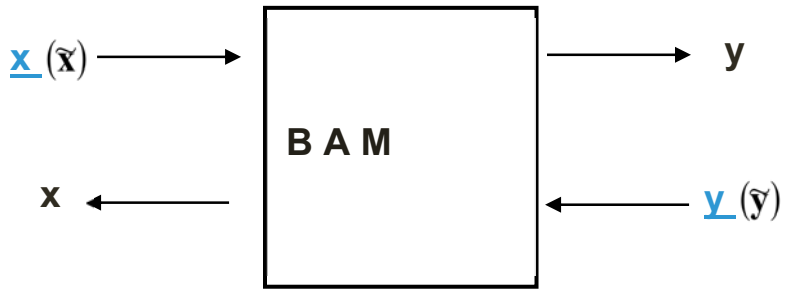


Figura 3.3.1 Esquema general de una Memoria Asociativa Bidireccional.

La BAM general es una "caja negra" que opera de la siguiente forma: dado un patrón **x** obtiene el patrón asociado **y**, y dado el patrón **y** obtiene el patrón asociado **x**. Además, si se asume que $\tilde{x}$ y $\tilde{y}$ son las versiones ruidosas de **x** y **y**, respectivamente, se espera que la BAM recupere los patrones correspondientes, **x** y **y**, libres de ruido.

Por ejemplo, el modelo de Kosko [18] podría esquematizarse como se muestra en la figura 3.3.2.

El modelo propuesto en este trabajo de tesis deberá comportarse de la misma forma que la BAM general en lo que respecta a la operación al recuperar patrones. El modelo se ha denominado Memorias Asociativas Bidireccionales Alfa-Beta (*Alpha-Beta BAM*) porque las memorias asociativas Alfa-Beta, tanto *max* como *min*, juegan un papel central en el diseño de este nuevo modelo.

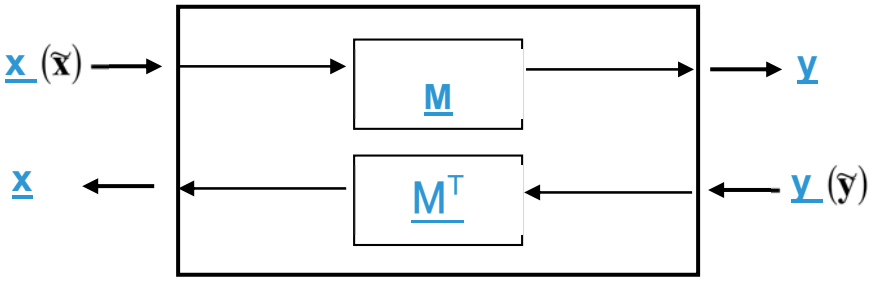


Figura 3.3.2 Modelo de Kosko según el esquema de la BAM general.

Dado que es una memoria asociativa bidireccional, la recuperación de patrones debe darse en dos direcciones opuestas. En cada una de las dos direcciones, el modelo consta de dos etapas, y las cuatro etapas se muestran en la figura 3.3.3.

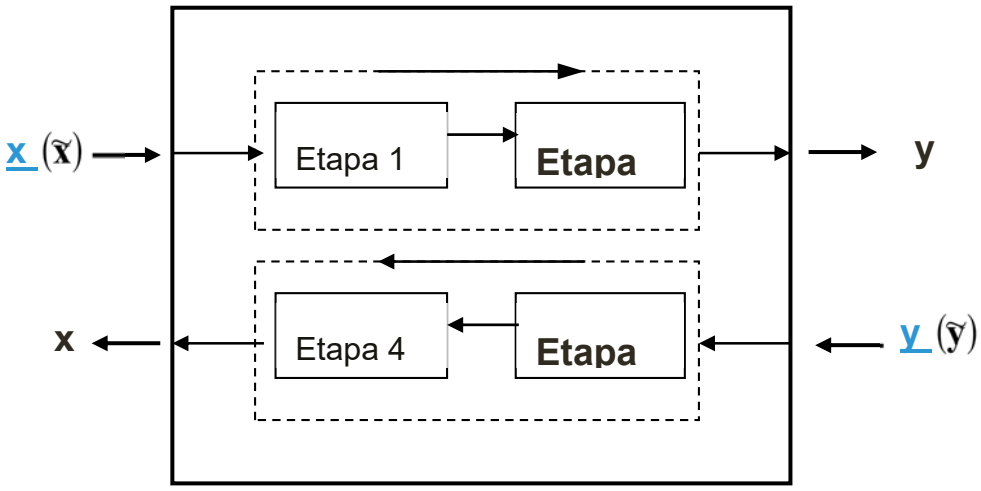


Figura 3.3.3 Esquema de las etapas de una memoria asociativa bidireccional Alfa-Beta.

A continuación se definen tres tipos de vectores que usados en este modelo., además, dos transformadas vectoriales originales, propias de las BAM Alfa-Beta.

<Definición 1 (One-Hot) *Sea el conjunto* $A = \{0, 1\}$ *y sean* $p \in \mathbf{Z}^+$, $p > 1$, $k \in \mathbf{Z}^+$, *tales que* $1 \leq k \leq p$. *El k-ésimo vector one-hot de p bits se define como el vector* $\mathbf{h}^k \in A^p$ *para el cual se cumple que la k-ésima componente* $h_k^k = 1$ *y las demás componentes* $h_j^k = 0$, $\forall j \neq k$, $1 \leq j \leq p$.

Nota *En esta definición se excluye el valor* $p = 1$ *porque un vector one-hot de dimensión 1, por su esencia misma, no tiene razón de ser.*

Definición 2 (Zero-Hot) *Sea el conjunto* $A = \{0, 1\}$ *y sean* $p \in \mathbf{Z}^+$, $p > 1$, $k \in \mathbf{Z}^+$, *tales que* $1 \leq k \leq p$. *El k-ésimo vector zero-hot de p bits se define como el vector* $\bar{\mathbf{h}}^k \in A^p$ *para el cual se cumple que la k-ésima componente* $\bar{h}_k^k = 0$ *y las demás componentes* $\bar{h}_j^k = 1$, $\forall j \neq k$, $1 \leq j \leq p$.

Nota *En esta definición se excluye el valor* $p = 1$ *porque un vector zero-hot de dimensión 1, por su esencia misma, no tiene razón de ser.*

Definición 3 (Transformada vectorial de expansión dimensional) *Sea el conjunto* $A = \{0, 1\}$ *y sean* $n \in \mathbf{Z}^+$, *y* $m \in \mathbf{Z}^+$. *Dados dos vectores cualesquiera* $\mathbf{x} \in A^n$ *y* $\mathbf{e} \in A^m$, *se define la transformada vectorial de expansión de orden m,* $\tau^e : A^n \rightarrow A^{n+m}$, *como* $\tau^e(\mathbf{x}, \mathbf{e}) = \mathbf{X} \in A^{n+m}$, *vector cuyas componentes son:* $X_i = x_i$ *para* $1 \leq i \leq n$ *y* $X_i = e_i$ *para* $n + 1 \leq i \leq n + m$.

Definición 4 (Transformada vectorial de contracción dimensional) *Sea el conjunto* $A = \{0, 1\}$ *y sean* $n \in \mathbf{Z}^+, m \in Z^+$ *tales que* $1 \leq m < n$. *Dado un vector cualesquiera* $\mathbf{X} \in A^{n+m}$,

se define la transformada vectorial de contracción de orden m, $\tau^c : A^{n+m} \rightarrow A^m$*, como* $\tau^c(\mathbf{X}, m) = \mathbf{c} \in A^m$*, vector cuyas componentes son:* $c_i = X_{i+n}$ *para* $1 \leq i < m$.

Definición 5 (Vector negado) *Sea el conjunto* $A = \{0, 1\}$ *y sea un vector* $\mathbf{s} \in A^n$*, se define el vector negado de* **s** *como el vector* $\bar{\mathbf{s}}$*, tal que* $\bar{s}_i = \neg s_i$*, donde* $\neg$ *es el operador lógico de negación booleano.*

Habiendo definido cinco conceptos importantes, se continúa con el proceso de la descripción del funcionamiento de las memorias asociativas bidireccionales Alfa-Beta.

En la dirección $\mathbf{x} \rightarrow \mathbf{y}$, las etapas 1 y 2 tienen como función proporcionar una $\mathbf{y}^k$ a la salida (k = 1, ..., p) dado un $\mathbf{x}^k$ a la entrada, asumiendo que en el conjunto fundamental se incluye la asociación ($\mathbf{x}^k$, $\mathbf{y}^k$), como se ilustra en la figura 3.3.5[8].

El modelo está diseñado de tal forma que la Etapa 2, constituida principalmente por el *Linear Associator* modificado, entregue como salida el vector $\mathbf{y}^k$; si recordamos que el *Linear Associator* tiene recuperación correcta cuando a la entrada se presentan vectores ortonormales, es deseable que a la salida de la Etapa 1 se tengan precisamente vectores de este tipo. En efecto, además de la Etapa 1 que es la contribución principal de esta tesis, otra contribución importante es la Etapa 2: en esta tesis se presentará una variación original del *Linear Associator* que permite obtener $\mathbf{y}^k$ a partir de un vector $\mathbf{h}^k$ *one-hot* en su k-ésima coordenada.

Por tanto, ya se tiene resuelta la tarea que debe realizar la Etapa 2. De la figura 4.4 se puede observar que falta descubrir qué hace la Etapa 1.

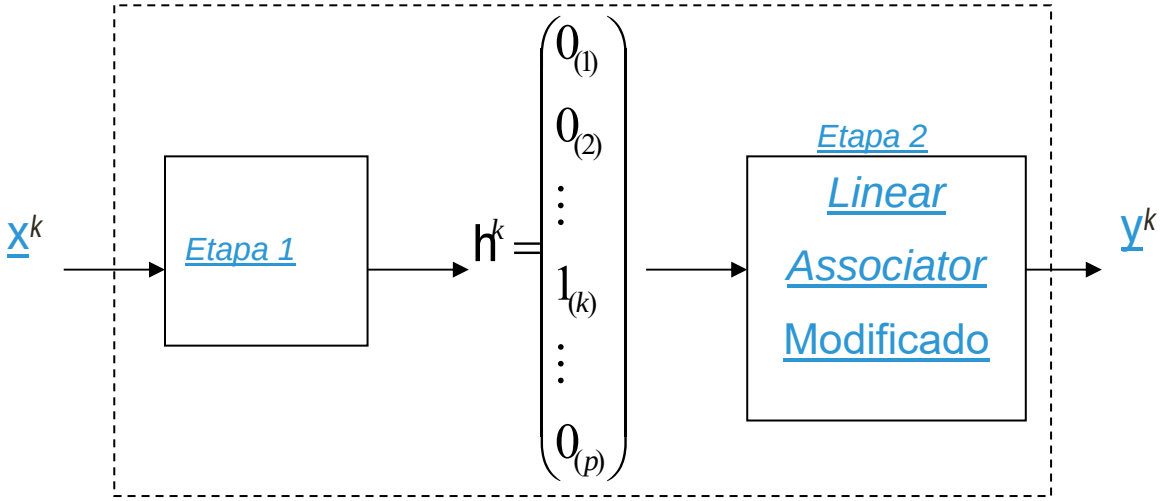


Figura 3.3.5 Esquema del proceso a realizar en el sentido **x** → **y**.

3.3.2 Algoritmo

En esta sección se describen paso a paso los procesos requeridos por la BAM Alfa-Beta tanto en la Fase de Aprendizaje, como en la Fase de Recuperación en el sentido de **x** → **y**, algoritmo para las Etapas 1 y 2, y en el sentido **y** → **x**, algoritmo para las Etapas 3 y 4.

3.3.2.1 Algoritmo de las Etapas 1 y 2

El siguiente algoritmo describe los pasos requeridos por la memoria asociativa bidireccional Alfa-Beta para realizar la fase de aprendizaje y la fase de recuperación en el sentido de **x** → **y**.

Paso 1. Como datos se tienen los *p* patrones de entrada **x**.

Paso 2. Se aplica la transformada de expansión vectorial del vector **x** para cada $\mathbf{X}^k$, utilizando como argumentos cada uno de los vectores de entrada $\mathbf{x}^k$ y cada vector *one-hot* $\mathbf{h}^k$.

Paso 3. Se aplica la transformada de expansión vectorial del vector **x** para cada $\overline{\mathbf{X}}^k$, utilizando como argumentos cada uno de los vectores de entrada $\mathbf{x}^k$ y cada vector *zero-hot* $\overline{\mathbf{h}}^k$.

Paso 4. Se crea una memoria asociativa Alfa-Beta *max* **V** con el conjunto fundamental

$$\left\{\left(\mathbf{X}^k, \mathbf{X}^k\right) \mid k = 1,..., p\right\}$$

Paso 5. Se crea una memoria asociativa Alfa-Beta *min* **Λ** con el conjunto fundamental

$$\left\{\left(\overline{\mathbf{X}}^k, \overline{\mathbf{X}}^k\right) \mid k = 1,..., p\right\}$$

Paso 6. Se crea una matriz **LAy**, que consiste de un *Linear Associator* modificado utilizando los patrones de salida **y**.

FASE DE RECUPERACIÓN

Paso 1. Presentar, a la entrada de la etapa 1, un vector del conjunto fundamental $\mathbf{x}^k \in A^n$ para algún índice $k \in \{1, ..., p\}$

Paso 2. Construir el vector $\mathbf{u} \in A^p$

Paso 3. Aplicar la transformada vectorial de expansión del vector **x** utilizando como argumentos el vector $\mathbf{x}^k$ y el vector **u**, para obtener **F**.

Paso 4. Operar la memoria autoasociativa Alfa-Beta *max* **V** con **F**, para obtener un vector **R.**

Paso 5. Aplicar la transformada de contracción del vector **R**, cuyos argumentos son el vector **R**, obtenido en el paso anterior, y *p* (número de patrones), para obtener el vector **r**.

Paso 6. Si r es un vector *one-hot*, entonces **r** es el *k-ésimo* vector *one-hot,* $\mathbf{h}^k$, (Basado en el Teorema 4.2) **entonces**.

Paso 6.1 Se realiza la operación $\mathbf{LAy} \cdot \mathbf{r}$, lo que resultará en el $\mathbf{y}^k$ correspondiente. **Fin**. **Si no,** entonces

Paso 7. Construir el vector $\mathbf{w} \in A^p$

Paso 8. Aplicar la transformada vectorial de expansión del vector **x** utilizando como argumentos el vector $\mathbf{x}^k$ y el vector **w**, para obtener **G**.

Paso 9. Operar la memoria autoasociativa Alfa-Beta *min* Λ con **G**, para obtener un vector **S.**

Paso 10. Aplicar la transformada de contracción del vector **S**, cuyos argumentos son el vector **S**, obtenido en el paso anterior, y *p* (número de patrones), para obtener el vector **s**.

Paso 11. Si s es un vector *zero-hot*, entonces **s** es el *k*-ésimo vector *zero-hot,* $\bar{\mathbf{h}}^k$, (Basado en el Teorema 4.4) **entonces**.

Paso 11.1 Se realiza la operación $\mathbf{LAy} \cdot \bar{\mathbf{s}}$, lo que resultará en el $\mathbf{y}^k$ correspondiente. **Fin**. **Si no,** entonces

Paso 12. Realizar la operación AND lógica entre r y $\bar{\mathbf{s}}$ para obtener el vector **t**, el cual (Basado en el Teorema 4.5) será igual al *k*-ésimo vector *one-hot*.

Paso 13. Realizar la operación $\mathbf{LAy} \cdot \mathbf{t}$, para obtener el $\mathbf{y}^k$ correspondiente. **Fin**.

3.3.5.2 Algoritmo de las Etapas 3 y 4

El siguiente algoritmo describe los pasos requeridos por la memoria asociativa bidireccional Alfa-Beta para realizar la fase de aprendizaje y la fase de recuperación en el sentido de $\mathbf{y} \rightarrow \mathbf{x}$.

FASE DE APRENDIZAJE

Paso 1. Como datos se tienen los *p* patrones de salida **y**.

Paso 2. Se aplica la transformada de expansión vectorial del vector **Y** para cada $\mathbf{Y}^k$, utilizando como argumentos cada uno de los vectores de entrada $\mathbf{y}^k$ y cada vector *one-hot* $\mathbf{h}^k$.

Paso 3. Se aplica la transformada de expansión vectorial del vector $\overline{\mathbf{Y}}$ para cada $\overline{\mathbf{Y}}^k$, utilizando como argumentos cada uno de los vectores de entrada $\mathbf{y}^k$ y cada vector *zero-hot* $\overline{\mathbf{h}}^k$.

Paso 4. Se crea una memoria asociativa Alfa-Beta *max* **V** con el conjunto fundamental

$$\left\{\left(\mathbf{Y}^k, \mathbf{Y}^k\right) \mid k = 1,..., p\right\}$$

Paso 5. Se crea una memoria asociativa Alfa-Beta *min* $\boldsymbol{\Lambda}$ con el conjunto fundamental

$$\left\{\left(\overline{\mathbf{Y}}^k, \overline{\mathbf{Y}}^k\right) \mid k = 1,..., p\right\}$$

Paso 6. Se crea una matriz **LAx**, que consiste de un *Linear Associator* modificado utilizando los patrones de entrada **x**.

FASE DE RECUPERACIÓN

Paso 1. Presentar, a la entrada de la etapa 1, un vector del conjunto fundamental $\mathbf{y}^k \in A^m$ para algún índice $k \in \{1, ..., p\}$

Paso 2. Construir el vector $\mathbf{u} \in A^p$

Paso 3. Aplicar la transformada vectorial de expansión del vector **y** utilizando como argumentos el vector $\mathbf{y}^k$ y el vector **u**, para obtener **F**.

Paso 4. Operar la memoria autoasociativa Alfa-Beta *max* **V** con **F**, para obtener un vector **R.**

Paso 5. Aplicar la transformada de contracción del vector **R**, cuyos argumentos son el vector **R**, obtenido en el paso anterior, y *p* (número de patrones), para obtener el vector **r**.

Paso 6. Si r es un vector *one-hot*, entonces **r** es el *k-ésimo* vector *one-hot,* $\mathbf{h}^k$, **entonces**.

Paso 6.1 Se realiza la operación $\mathbf{LAx} \cdot \mathbf{r}$, lo que resultará en el $\mathbf{x}^k$ correspondiente. **Fin**. **Si no,** entonces

Paso 7. Construir el vector $\mathbf{w} \in A^p$

Paso 8. Aplicar la transformada vectorial de expansión del vector **y** utilizando como argumentos el vector $\mathbf{y}^k$ y el vector **w**, para obtener **G**.

Paso 9. Operar la memoria autoasociativa Alfa-Beta *min* Λ con **G**, para obtener un vector **S.**

Paso 10. Aplicar la transformada de contracción del vector **S**, cuyos argumentos son el vector **S**, obtenido en el paso anterior, y *p* (número de patrones), para obtener el vector **s**.

Paso 11. Si s es un vector *zero-hot*, entonces **s** es el *k*-ésimo vector *zero-hot,* $\overline{\mathbf{h}}^k$, **entonces**.

Paso 11.1 Se realiza la operación $\mathbf{LAx} \cdot \overline{\mathbf{s}}$, lo que resultará en el $\mathbf{x}^k$ correspondiente. **Fin**. **Si no,** entonces

Paso 12. Realizar la operación AND lógica entre r y $\bar{\mathbf{s}}$ para obtener el vector **t**, el cual será igual al *k*-ésimo vector *one-hot*.

Paso 13. Realizar la operación **LAx** · **t**, para obtener el $\mathbf{x}^k$ correspondiente. **Fin**.

3.4 Ejemplo demostrativo del funcionamiento de la BAM Alfa-Beta.

Para efecto de comparación de la eficiencia de nuestro proyecto, a continuación de ilustra un ejemplo demostrativo del funcionamiento de la BAM Alfa-Beta.

Sean los siguientes 4 pares de patrones (*p*=4), los patrones de entrada, **x** con dimensión 4 (*n*=4) y los patrones de salida, **y** con dimensión 3 (*m*=3).

$$\mathbf{x}^1=\begin{bmatrix}1\\1\\1\\1\end{bmatrix},\ \mathbf{y}^1=\begin{bmatrix}1\\0\\1\end{bmatrix},\ \mathbf{x}^2=\begin{bmatrix}0\\0\\0\\0\end{bmatrix},\ \mathbf{y}^2=\begin{bmatrix}1\\0\\0\end{bmatrix},\ \mathbf{x}^3=\begin{bmatrix}0\\0\\1\\0\end{bmatrix},\ \mathbf{y}^3=\begin{bmatrix}0\\1\\0\end{bmatrix},\ \mathbf{x}^4=\begin{bmatrix}1\\1\\0\\0\end{bmatrix},\ \mathbf{y}^4=\begin{bmatrix}1\\1\\0\end{bmatrix}$$

FASE DE APRENDIZAJE

- Se aplica la transformada vectorial de expansión del vector **x**, para obtener cada uno de los vectores $\mathbf{X}^k$ y $\overline{\mathbf{X}}^k$

$$\boxed{\mathbf{X}=\tau^e(\mathbf{x},\mathbf{h})} \qquad \boxed{\overline{\mathbf{X}}=\tau^e(\mathbf{x},\overline{\mathbf{h}})}$$

$$\mathbf{X}^1 = \begin{bmatrix}1\\1\\1\\1\\1\\0\\0\\0\end{bmatrix} \quad \mathbf{X}^2 = \begin{bmatrix}0\\0\\0\\0\\0\\1\\0\\0\end{bmatrix} \quad \mathbf{X}^3 = \begin{bmatrix}0\\0\\1\\0\\0\\0\\1\\0\end{bmatrix} \quad \mathbf{X}^4 = \begin{bmatrix}1\\1\\0\\0\\0\\0\\0\\1\end{bmatrix}$$

Vectores *one-hot*

$$\overline{\mathbf{X}}^1 = \begin{bmatrix}1\\1\\1\\1\\0\\1\\1\\1\end{bmatrix} \quad \overline{\mathbf{X}}^2 = \begin{bmatrix}0\\0\\0\\0\\1\\0\\1\\1\end{bmatrix} \quad \overline{\mathbf{X}}^3 = \begin{bmatrix}0\\0\\1\\0\\1\\1\\0\\1\end{bmatrix} \quad \overline{\mathbf{X}}^4 = \begin{bmatrix}1\\1\\0\\0\\1\\1\\1\\0\end{bmatrix}$$

Vectores *zero-hot*

- Se generan las memorias autoasociativas Alfa-Beta *max* $\mathbf{V} = \bigvee_{k=1}^{4} \left(\mathbf{X}^k \otimes \left(\mathbf{X}^k \right)^t \right)$ y *min* $\mathbf{\Lambda} = \bigwedge_{k=1}^{4} \left(\overline{\mathbf{X}}^k \otimes \left(\overline{\mathbf{X}}^k \right)^t \right)$

$$\mathbf{V} = \begin{bmatrix}1&1&2&2&2&2&2&1\\1&1&2&2&2&2&2&2\\2&2&1&2&2&2&2&2\\1&1&1&1&1&2&2&2\\1&1&1&1&1&2&2&2\\2&2&2&2&2&1&2&2\\2&2&1&2&2&2&1&2\\1&1&2&2&2&2&2&1\end{bmatrix} \quad \text{y} \quad \mathbf{\Lambda} = \begin{bmatrix}1&1&0&1&0&0&0&0\\1&1&0&1&0&0&0&0\\0&0&1&1&0&0&0&0\\0&0&0&1&0&0&0&0\\0&0&0&0&1&0&0&0\\1&1&1&1&0&1&0&0\\1&1&0&1&0&0&1&0\\0&0&1&1&0&0&0&1\end{bmatrix}$$

- Se crea el *Linear Associator* Modificado utilizando los patrones de salida **y**

$$\mathbf{LAy} = \begin{bmatrix}1&1&0&1\\0&0&1&1\\1&0&0&0\end{bmatrix}$$

FASE DE RECUPERACIÓN

- Se presenta a la entrada de la BAM Alfa-Beta el patrón $\mathbf{x}^3$, se construye el vector **u** y se construye la expansión

$$\mathbf{x}^3 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \qquad \mathbf{u} = \sum_{i=1}^{4} \mathbf{h}^i = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$$\mathbf{F} = \tau^e(\mathbf{x}^3, \mathbf{u}) \longrightarrow \mathbf{F} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

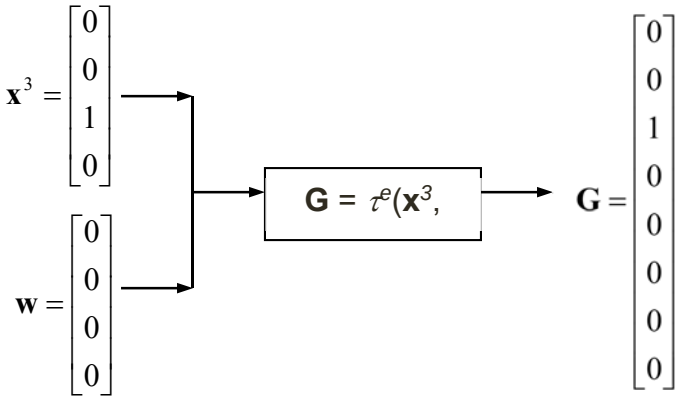
- Se opera la memoria autoasociativa Alfa-Beta *max* **V** con **F**, para obtener **R.**

$$\mathbf{R} = \mathbf{V} \Delta_\beta \mathbf{F} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

- Se realiza la contracción: $\mathbf{r} = \tau^c(\mathbf{R}, 4) = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$

Debido a que **r** no es un vector *one-hot*, entonces se continúa con el algoritmo.

- Se construye el vector **u** y con el patrón $\mathbf{x}^3$, se construye la expansión, obteniendo **G**.



- Se opera la memoria autoasociativa Alfa-Beta *min* $\boldsymbol{\Lambda}$ con **G**

$$\mathbf{S} = \boldsymbol{\Lambda}\nabla_{\beta}\mathbf{G} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

- Se realiza la contracción $\mathbf{s} = \tau^{c}(\mathbf{S},4) = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}$

El vector **s** no es un vector *zero-hot*, por lo que se continúa con el proceso.

- Se realiza la operación AND entre el vector **r** y el vector negado de **s**, $\bar{s}$, para obtener el vector **t.**

$$\mathbf{t} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \Lambda \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

Con esta operación se obtiene el tercer vector *one-hot*

- Se obtiene el vector $\mathbf{y}^3$ mediante

$$\mathbf{y}_{rec} = \mathbf{LAy} \cdot \mathbf{t} = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \mathbf{y}^3$$

Como se puede observar, el patrón de salida y^3 fue recuperado correctamente, por lo que queda demostrada la eficiencia de la memoria asociativa bidireccional Alfa-Beta.

SOY POLITÉCNICO

Para alcanzar las conquistas universales y ofrecerlas a mí pueblo

CAPÍTULO 4

DESARROLLO DEL PROYECTO

Como se mencionó anteriormente en el capítulo 2, este proyecto está basado principalmente en el artículo de Tasuya Sakato, Motoyuki Ozeki y Natsuki Oka titulado *"A Computacional Model of Imitation and Autonomous Behavior in Continuous Spaces""* [7], el cual fue publicado en el año 2013; y donde se utilizan dos agentes uno de aprendizaje y otro óptimo. El agente de aprendizaje realiza diversas acciones en el cual un agente de aprendizaje realiza acciones de acuerdo a la imitación del agente óptimo, a través de módulos de aprendizaje por refuerzo (RL) y módulos de imitación.

Su entorno se sitúa en la simulación de una mesa de comedor, en donde se colocan tres objetos a identificar: una piedra, un racimo de plátanos y una manzana. El objetivo principal es que el agente de aprendizaje observe las acciones que ejecuta el agente óptimo y ejecute la acción correcta.

Para cada uno de los objetos se espera una respuesta adecuada por parte del agente de aprendizaje, es decir, si el objeto tomado de la mesa es una piedra, la acción que se espera ejecute dicho agente será arrojar dicho objeto; si se trata de el racimo de plátanos o la manzana, se espera que la acción elegida sea comer dichos objetos.

Se ejecuta un turno a la vez, el agente óptimo tomara un objeto de la mesa y ejecutara la acción correcta, mientras tanto, el agente de aprendizaje observara los movimientos y respuesta de este, y decidirá cuál de los dos métodos de aprendizaje es el conveniente: aprendizaje por refuerzo, o aprendizaje por imitación, dependiendo de las recompensas y resultados que obtuvo el agente óptimo.

El entorno simulado por Sakato, Ozeki y Oka [7], para dicho proyecto se ilustra en la figura 4.1, en ella se puede observar al agente óptimo situado en la parte de abajo, y al agente de aprendizaje que se ubica en la parte superior, a su vez se observa la mesa de comedor donde se sitúan cada uno de los tres objetos utilizados para las pruebas de aprendizaje.

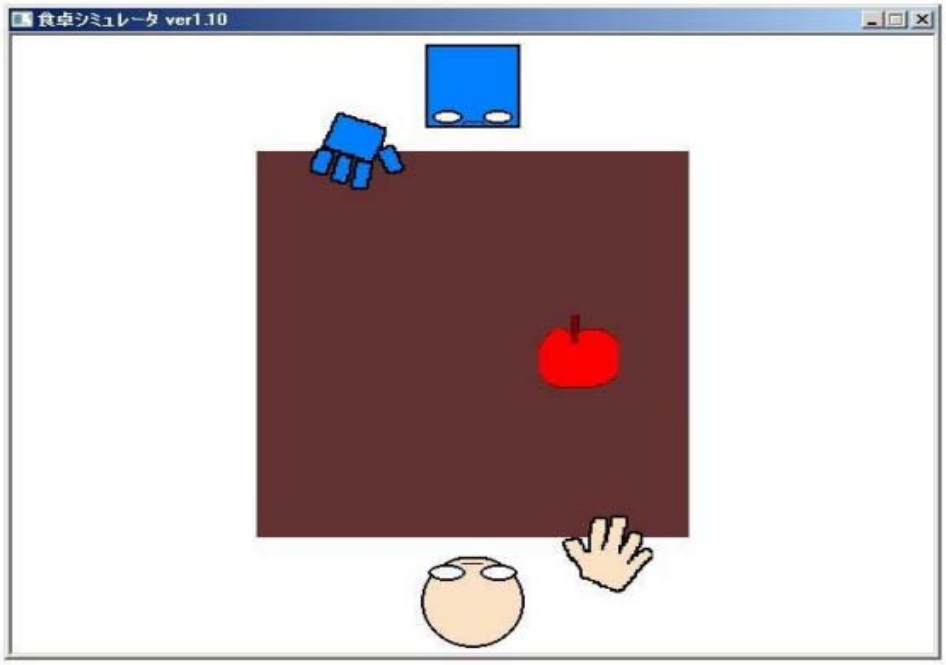


Figura 4.1 Entorno de Aprendizaje [7]

4.1 Implementación con BAM Alfa-Beta

Ya en el capítulo 2 se describió a detalle el modelo propuesto por Sakato T., Ozeki M., y Oka N.

Sin embargo, el objetivo principal de este trabajo es lograr el correcto aprendizaje de nuestro agente, usando las BAM Alfa-Beta, reduciendo con ello el tiempo de

aprendizaje de este, y a su vez aumentando el número de elementos a identificar en su entorno, incrementando con ello la cantidad de acciones a ejecutar por este.

El proyecto se desarrollara a nivel software, creando patrones de entrada por cada elemento a identificar, llamando a estos x^1, x^2,..... x^n. Mientras que los patrones de salida serán las acciones a realizar identificadas como y^1,y^2,...y^n.

4.2 Elementos a identificar y sus respectivas características

Se tomó la decisión de aumentar el número de elementos comestibles (frutas), y no comestibles (objetos), teniendo un total de 10 frutas y 5 objetos, cada uno con su respectivo patrón de salida, es decir la acción esperada.

LISTADO DE FRUTAS A IDENTIFICAR
Uva morada
Uva verde
Mango
Naranja
Guayaba
Manzana roja
Manzana verde
Plátano
Fresa
Pera

Tabla 4.1 Lista de las 10 frutas a identificar.

LISTADO DE OBJETOS
Libro azul
Libro rojo
Ladrillo
Piedra
Vaso
Lápiz

Tabla 4.2 lista de los cinco objetos a identificar

Creando con ello dos tipos diferentes de clases como se ilustra en la tabla 4.1.

CLASE	
10	FRUTA
01	OBJETO

Tabla 4.3 Clases creadas.

Teniendo cada uno de los elementos de estas clases sus propias características, como son color y forma.

FORMA	COLOR
CÍRCULO	ROJO
RECTÁNGULO	NARANJA
OVALO	AMARILLO
CILINDRO	VERDE
OTRO	MORADA
	CAFÉ
	AZUL
	GRIS

Tabla 4.4 Formas y colores a identificar

El agente inteligente deberá asociar las características anteriores a cada una de las frutas y objetos, para después proceder a ejecutar la acción correcta dependiendo del elemento identificado.

La clase fruta deberá quedar asociada de la siguiente forma:

OVALO/COLOR	CIRCULO/COLOR	OTRO/COLOR
Uva /Morado, Verde	Naranja/ Naranja	Plátano/Amarillo
Durazno/Naranja	Guayaba/Amarillo	Fresa/Rojo
Mamey/Café	Manzana/Rojo, Verde	Pera/Verde
Mango/Amarillo		

Tabla 4.5 Elementos de la clase fruta, identificados por su forma y color.

Las características de la clase objeto se muestran en la tabla 4.4.

RECTÁNGULO/COLOR	CILINDRO/COLOR	OTRO/COLOR
Libro/Azul Ladrillo/Rojo	Vaso/Azul Lápiz/Amarillo	Piedra/Gris

Tabla 4.6 Elementos de la clase objeto identificados por su forma y color.

Como se puede observar en las tablas 4.3 y 4.4, cada una de las frutas y objetos posee diferentes formas y colores, por lo que el agente inteligente deberá asociar cada uno de ellos a la acción correcta a ejecutar.

Las acciones esperadas para cada uno de los objetos se ilustra en la tabla 4.7:

ACCIÓN ESPERADA						
Comer	Uva	Durazno	Guayaba	Manzana	Fresa	Pera
Arrojar	Piedra	Ladrillo				
Pelar	Mango	Naranja	Plátano			
Abrir	Libro					
Beber	Vaso					
Escribir	Lápiz					

Tabla 4.7 Acciones a realizar según el objeto o fruta identificado.

4.3 Diseño de los Patrones de Entrada

Una vez teniendo las características de cada fruta y objeto a identificar, se procede a la asignación de los bits correspondientes a los 16 patrones de entrada, estos dependerán de la clase a la cual corresponde cada uno, la forma de estos y su respectivo color.

En la tabla 4.8 se pueden observar los patrones de entrada correspondientes a cada fruta y objeto a identificar.

ELEMENTO	*CONJUNTO DE BITS*
Uva morada	1 0 0 1 1 1 0 1
Uva verde	1 0 0 1 1 1 0 0
Mamey	1 0 0 1 1 1 1 0
Mango	1 0 0 1 1 0 1 1
Naranja	1 0 0 0 1 0 1 0
Guayaba	1 0 0 0 1 0 1 1
Manzana roja	1 0 0 0 1 0 0 1
Manzana verde	1 0 0 0 1 1 0 0
Plátano	1 0 0 0 0 0 1 1
Fresa	1 0 0 0 0 0 0 1
Pera	1 0 0 0 0 1 0 0
Libro azul	0 1 0 1 0 1 1 1
Ladrillo	0 1 0 1 0 0 0 1
Piedra	0 1 0 0 0 0 0 0
Vaso	0 1 1 0 0 1 1 1
Lápiz	0 1 1 0 0 0 1 1

Tabla 4.8 Conjunto de elementos y sus respectivos bits

Una vez creados los patrones de entrada se procede a la creación del algoritmo a ejecutar por la BAM Alfa-Beta.

4.4 Pruebas realizadas

A continuación, se analizara el proceso de recuperación de la Alfa-Beta con los 16 patrones de entrada. Para demostrar el proceso de aprendizaje y recuperación, se ilustraran 4 de los 16 patrones de entrada.

Sean los siguientes 4 pares de patrones (p=4), los patrones de entrada, $\mathbf{x}$ con dimensión 8 (n=8).

$$\mathbf{x}^1 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 1 \end{bmatrix} \ldots \ \mathbf{x}^4 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \end{bmatrix} \ldots \ \mathbf{x}^{14} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \ldots\ldots \ \mathbf{x}^{16} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

FASE DE APRENDIZAJE

- Se aplica la transformada vectorial de expansión del vector $\mathbf{x}$, para obtener cada uno de los vectores $\mathbf{X}^k$ y $\overline{\mathbf{X}}^k$

$$\boxed{\mathbf{X} = \tau^e(\mathbf{x}, \mathbf{h})} \qquad \boxed{\overline{\mathbf{X}} = \tau^e(\mathbf{x}, \overline{\mathbf{h}})}$$

$$\mathbf{X}^1 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \mathbf{X}^4 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \mathbf{X}^{14} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \quad \mathbf{X}^{16} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Vectores *one-hot*

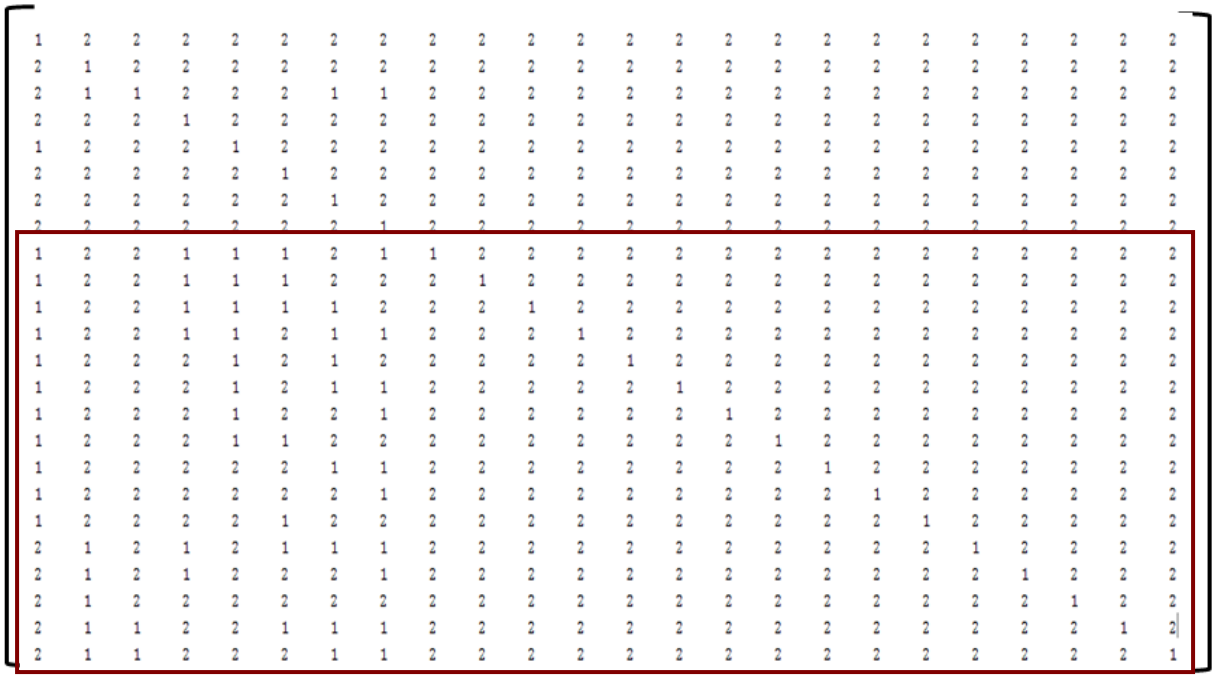
$$\overline{\mathbf{X}}^1 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \quad \overline{\mathbf{X}}^4 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \quad \overline{\mathbf{X}}^{14} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \end{bmatrix} \quad \overline{\mathbf{X}}^{16} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Vectores *zero-hot*

- Se generan las memorias autoasociativas Alfa-Beta *max* $\mathbf{V} = \bigvee_{k=1}^{16} \left(\mathbf{X}^k \otimes \left(\mathbf{X}^k \right)^t \right)$ y *min* $\mathbf{\Lambda} = \bigwedge_{k=1}^{16} \left(\overline{\mathbf{X}}^k \otimes \left(\overline{\mathbf{X}}^k \right)^t \right)$.

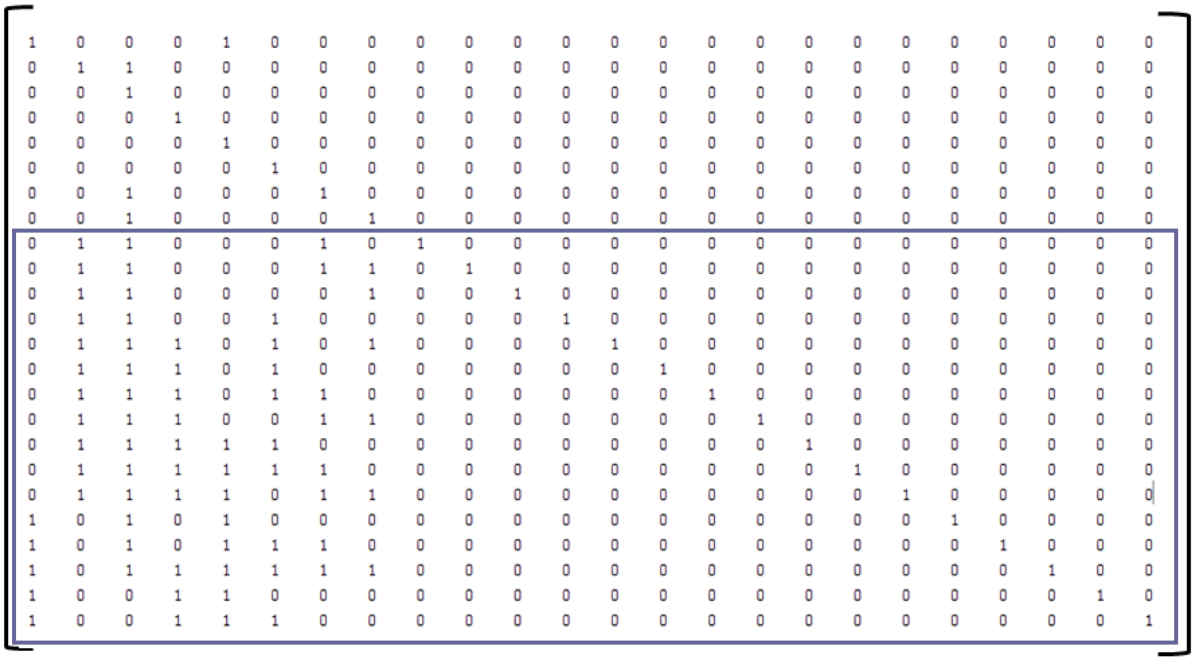
Memoria autoasociativa Alfa-Beta max de los 16 patrones es:

$\mathbf{V} =$



Memoria autoasociativa Alfa-Beta min:

$\mathbf{\Lambda} =$



- Se crea el *Linear Associator* Modificado utilizando los patrones de salida **y.**

$$\mathbf{LAy}_{\cdot} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

FASE DE RECUPERACIÓN PARA EL PATRÓN $\boldsymbol{x^1}$

- Se presenta a la entrada de la BAM Alfa-Beta el patrón $\mathbf{x}^1$, se construye el vector **u** y se construye la expansión

$$\mathbf{x}^1 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 1 \end{bmatrix} \qquad \mathbf{u} = \sum_{i=1}^{8} \mathbf{h}^i = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$$\mathbf{F} = \tau^e(\mathbf{x}^1, \mathbf{u}) \rightarrow \mathbf{F} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

- Se opera la memoria autoasociativa Alfa-Beta *max* **V** con **F**

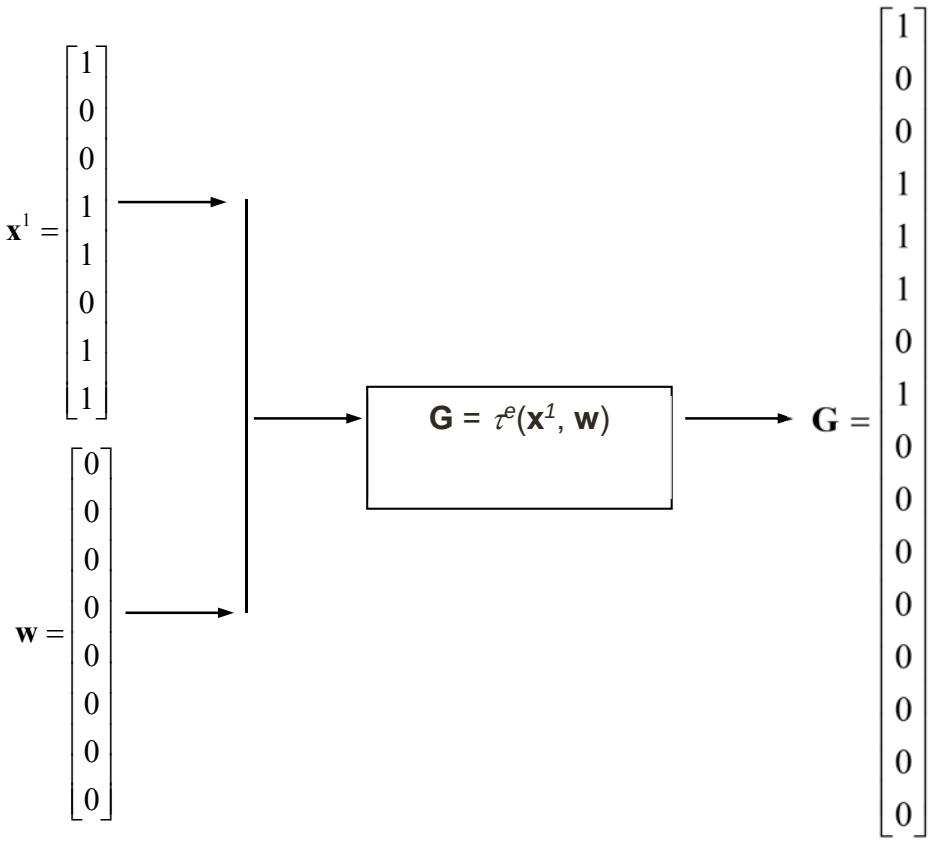
$$\mathbf{R} = \mathbf{V}\Delta_{\beta}\mathbf{F} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Se realiza la contracción:

$$\mathbf{r} = \tau^{\epsilon}(\mathbf{R}, \delta) = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Debido a que **r** no es un vector *one-hot*, entonces se continúa con el algoritmo

- Se construye el vector **u** y con el patrón $\mathbf{x}^1$, se construye la expansión



- Se opera la memoria autoasociativa Alfa-Beta *min* Λ con **G**

$$\mathbf{S} = \Lambda\nabla_{\beta}\mathbf{G} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

Se realiza la contracción:

$$\mathbf{s} = \tau^{c}(\mathbf{S},8) = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

El vector **s** no es un vector *zero-hot*, por lo que se continúa con el proceso.

- Se realiza la operación AND entre el vector **r** y el vector negado de **s**, $\bar{s}$, para obtener el vector **t**

$$\mathbf{t} = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \Lambda \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Con esta operación se obtiene el cuarto vector *one-hot*

- Se obtiene el vector $\mathbf{y}^1$ mediante

$$\mathbf{y}_{rec} = \mathbf{LAy} \cdot \mathbf{t} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \mathbf{y}^1$$

FASE DE RECUPERACIÓN PARA EL PATRÓN x^4

- Se presenta a la entrada de la BAM Alfa-Beta el patrón $\mathbf{x}^4$, se construye el vector **u** y se construye la expansión

$$\mathbf{x}^4 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \end{bmatrix} \qquad \mathbf{u} = \sum_{i=1}^{8} \mathbf{h}^i = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$$\mathbf{F} = \tau^e(\mathbf{x}^4, \mathbf{u})$$

$$\mathbf{F} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

- Se opera la memoria autoasociativa Alfa-Beta max **V** con **F**

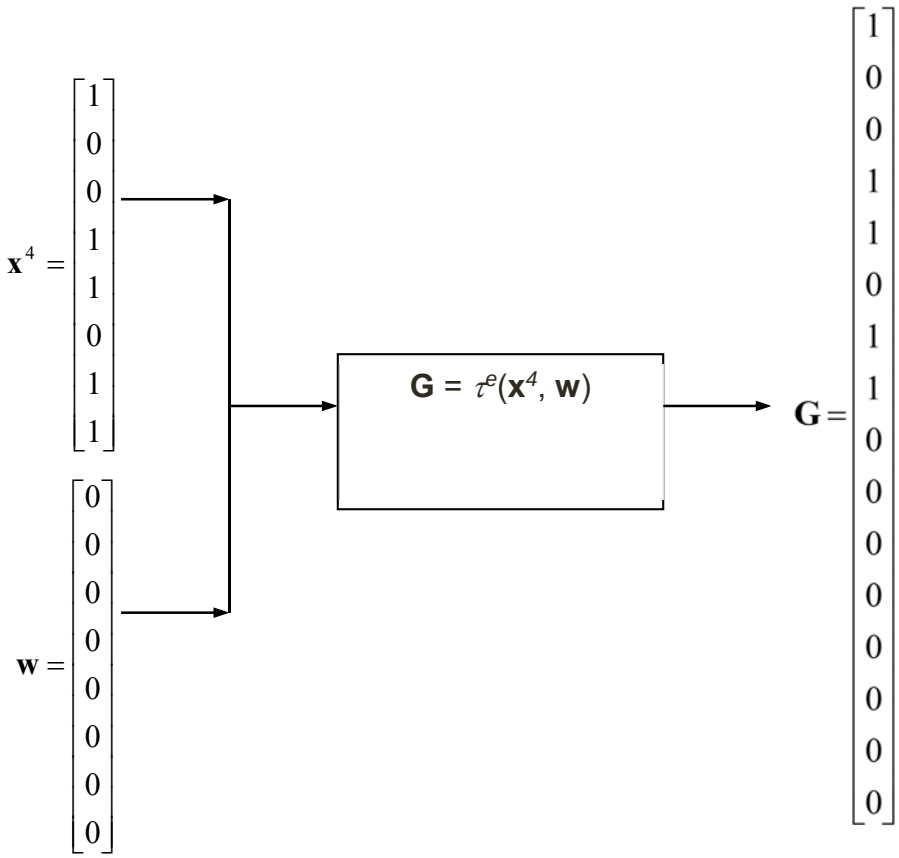
$$\mathbf{R} = \mathbf{V}\Delta_{\beta}\mathbf{F} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Se realiza la contracción:

$$\mathbf{r} = \tau^{c}(\mathbf{R},8) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Debido a que **r** no es un vector *one-hot*, entonces se continúa con el algoritmo

- Se construye el vector **u** y con el patrón $\mathbf{x}^4$, se construye la expansión



- Se opera la memoria autoasociativa Alfa-Beta *min* Λ con **G**

$$\mathbf{S} = \mathbf{\Lambda}\nabla_{\beta}\mathbf{G} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

Se realiza la contracción :

$$\mathbf{s} = \tau^{c}(\mathbf{S},8) = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

El vector **s** no es un vector *zero-hot*, por lo que se continúa con el proceso.

- Se realiza la operación AND entre el vector **r** y el vector negado de **s**, $\bar{s}$, para obtener el vector **t**

$$\mathbf{t} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \Lambda \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

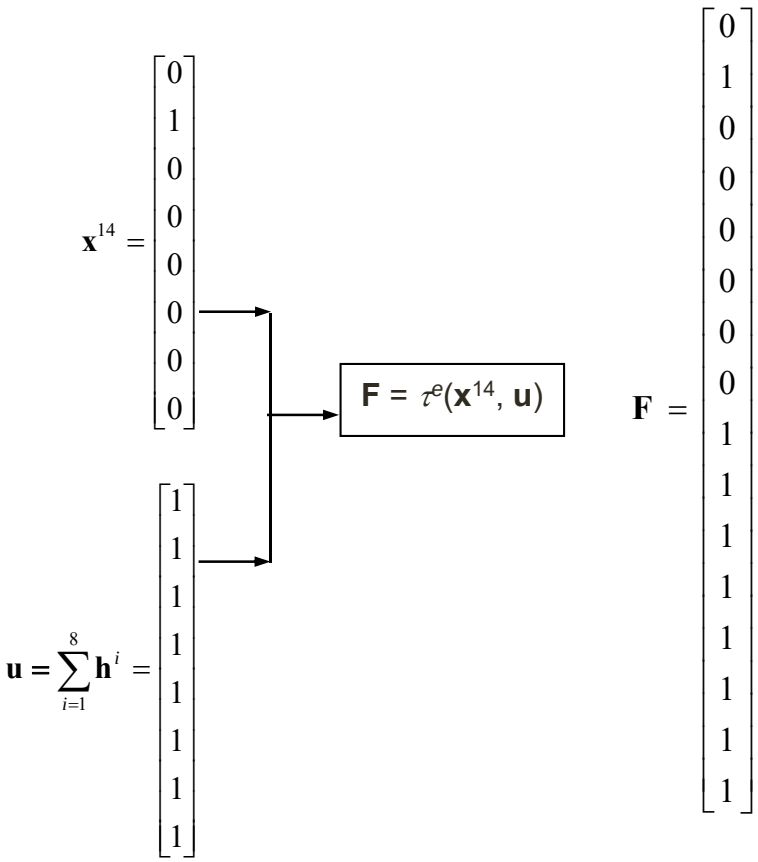
Con esta operación se obtiene el cuarto vector *one-hot*

- Se obtiene el vector $\mathbf{y}^4$ mediante

$$\mathbf{y}_{rec} = \mathbf{LAy} \cdot \mathbf{t} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \mathbf{y}^4$$

FASE DE RECUPERACIÓN PARA EL PATRÓN x^{14}

- Se presenta a la entrada de la BAM Alfa-Beta el patrón $\mathbf{x}^{14}$, se construye el vector **u** y se construye la expansión



- Se opera la memoria autoasociativa Alfa-Beta *max* $\mathbf{V}$ con $\mathbf{F}$

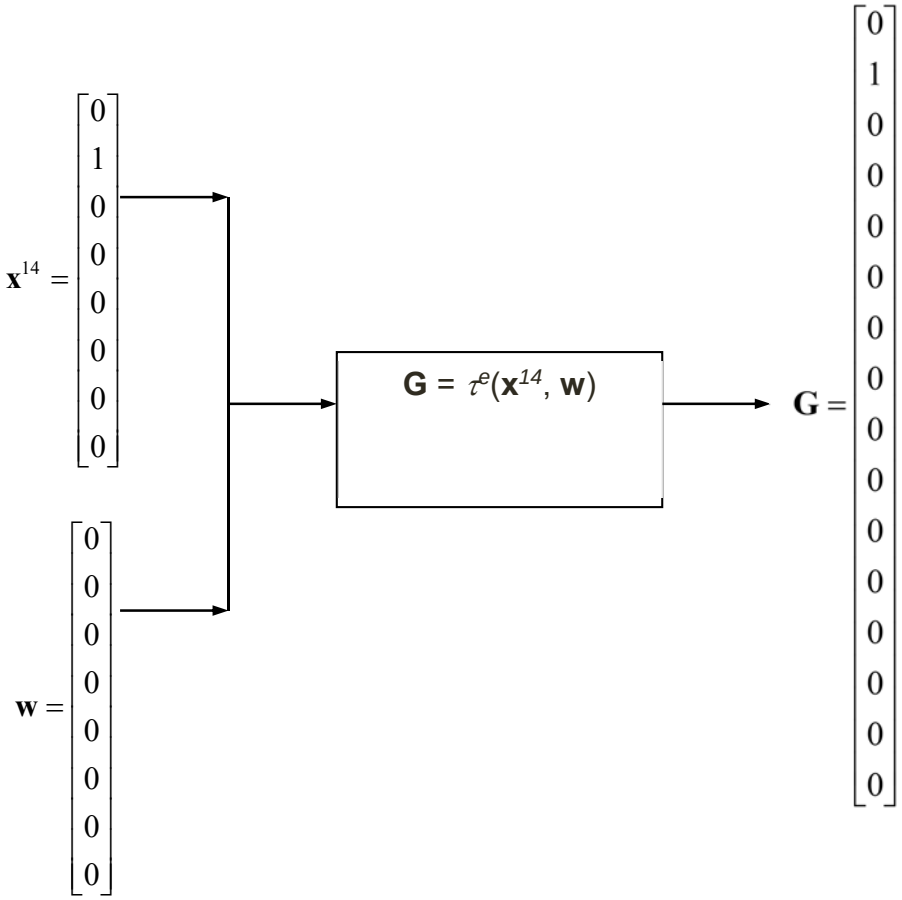
$$\mathbf{R} = \mathbf{V}\Delta_{\beta}\mathbf{F} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

Se realiza la contracción :

$$\mathbf{r} = \tau^{c}(\mathbf{R},8) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

Debido a que **r** no es un vector *one-hot*, entonces se continúa con el algoritmo

- Se construye el vector **u** y con el patrón $\mathbf{x}^{14}$, se construye la expansión



- Se opera la memoria autoasociativa Alfa-Beta *min* Λ con **G**

$$\mathbf{S} = \mathbf{\Lambda}\nabla_{\beta}\mathbf{G} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Se realiza la contracción :

$$\mathbf{s} = \tau^{c}(\mathbf{S},8) = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

El vector **s** no es un vector *zero-hot*, por lo que se continúa con el proceso.

- Se realiza la operación AND entre el vector **r** y el vector negado de **s**, $\bar{s}$, para obtener el vector **t**

$$\mathbf{t} = \begin{bmatrix}0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\1\\0\\0\end{bmatrix} \Lambda \begin{bmatrix}0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\1\\1\\1\\1\\1\\1\end{bmatrix} = \begin{bmatrix}0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\1\\0\\0\end{bmatrix}$$

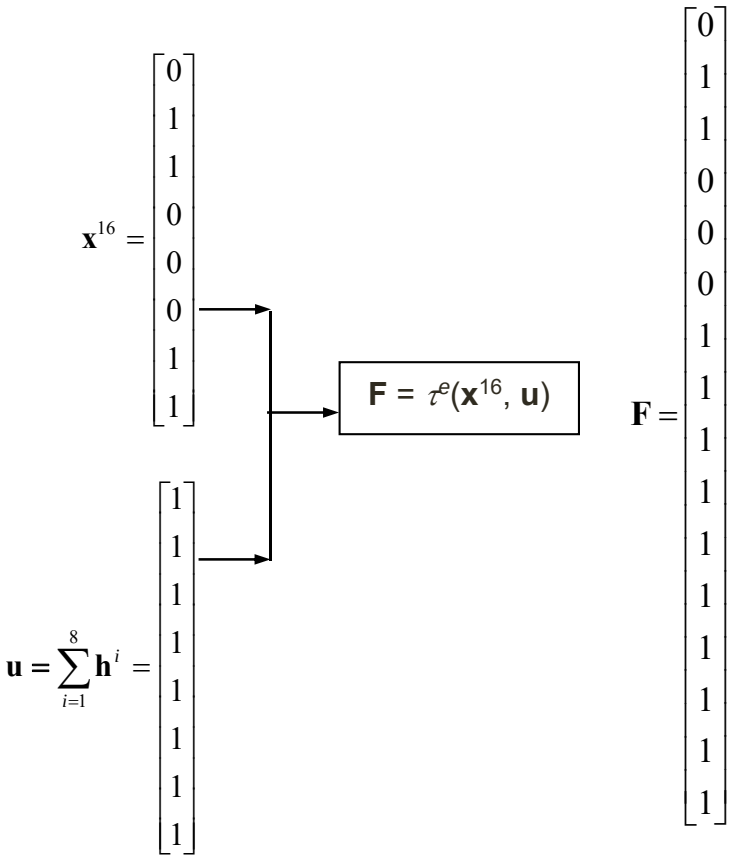
Con esta operación se obtiene el cuarto vector *one-hot*

- Se obtiene el vector $\mathbf{y}^{14}$ mediante

$$\mathbf{y}_{rec} = \mathbf{LAy} \cdot \mathbf{t} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \mathbf{y}^{14}$$

FASE DE RECUPERACIÓN PARA EL PATRÓN $\boldsymbol{x^{16}}$

- Se presenta a la entrada de la BAM Alfa-Beta el patrón $\mathbf{x}^{16}$, se construye el vector **u** y se construye la expansión



- Se opera la memoria autoasociativa Alfa-Beta *max* **V** con **F**

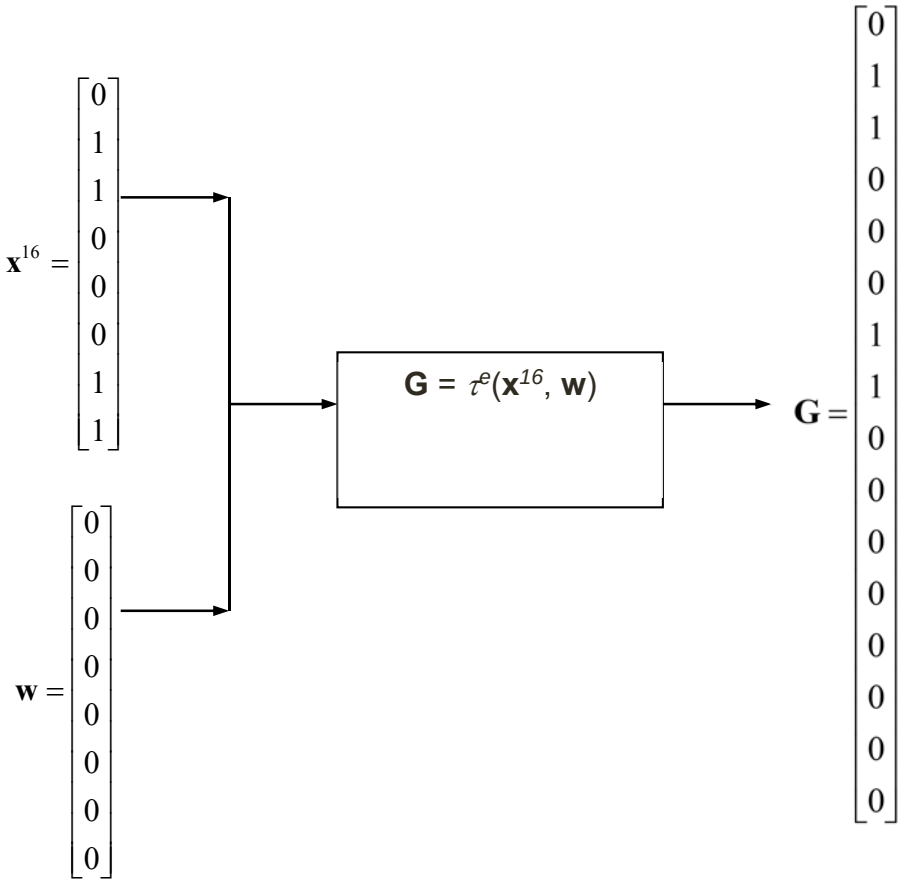
$$\mathbf{R} = \mathbf{V}\Delta_{\beta}\mathbf{F} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

Se realiza la contracción :

$$\mathbf{r} = \tau^{c}(\mathbf{R},8) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

Debido a que **r** no es un vector *one-hot*, entonces se continúa con el algoritmo

- Se construye el vector **u** y con el patrón $\mathbf{x}^{16}$, se construye la expansión



- Se opera la memoria autoasociativa Alfa-Beta *min* $\boldsymbol{\Lambda}$ con **G**

$$\mathbf{S} = \mathbf{\Lambda}\nabla_{\beta}\mathbf{G} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

Se realiza la contracción:

$$\mathbf{s} = \tau^{c}(\mathbf{S},8) = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

El vector **s** no es un vector *zero-hot*, por lo que se continúa con el proceso.

- Se realiza la operación AND entre el vector **r** y el vector negado de **s**, $\bar{s}$, para obtener el vector **t**

$$\mathbf{t} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} \Lambda \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Con esta operación se obtiene el cuarto vector *one-hot*

- Se obtiene el vector $\mathbf{y}^{16}$ mediante

$$\mathbf{y}_{rec} = \mathbf{LAy} \cdot \mathbf{t} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \end{bmatrix} = \mathbf{y}^{16}$$

Como se pudo observar, los patrones de entrada fueron correctamente recuperados y asociados con su correspondiente acción.

En el siguiente capítulo se ilustran los resultados obtenidos con cada uno de los 16 patrones de entrada.

SOY POLITÉCNICO

Porque me duele la patria en mis entrañas y aspiro a calmar sus dolencias

CAPÍTULO 5

RESULTADOS

5.1 Comprobación de Resultados

En el capítulo anterior se ilustro de manera teórica la correcta recuperación de cuatro de los 16 patrones de entrada, por lo que a continuación se ilustrara el resultado obtenido para cada uno de los 16 patrones evaluados por la memoria asociativa.

Al abrir nuestro programa y darle F5 para ejecutarlo, nos pide ingresemos el objeto seleccionado (es decir el patrón a evaluar), para el siguiente ejemplo se tomara el patrón número uno, correspondiente a uva morada, por lo que la memoria deberá asociar dicho objeto a la respuesta esperada, para ser más ilustrativo el procedimiento operado por la memoria asociativa, la respuesta esperada (es decir nuestro patrón de salida), fue asociada a una imagen que representa la acción esperada.

Ejemplo:

Al ejecutar nuestro programa y elegir como objeto seleccionado el patrón x^1 la acción que el agente ejecutara se ilustrara a través de la figura 1, la cuál será comer:

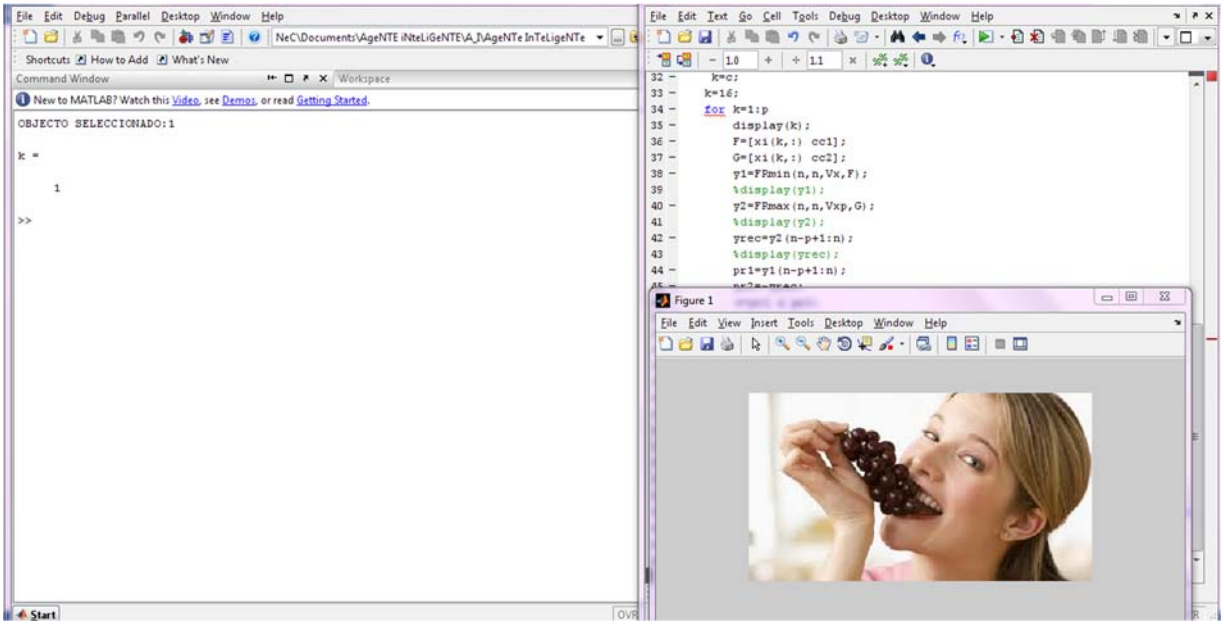


Figura 5.1 Respuesta para el patrón x^1.

Al evaluar el patrón x^2 correspondiente a uvas verdes la acción asociada es comer:

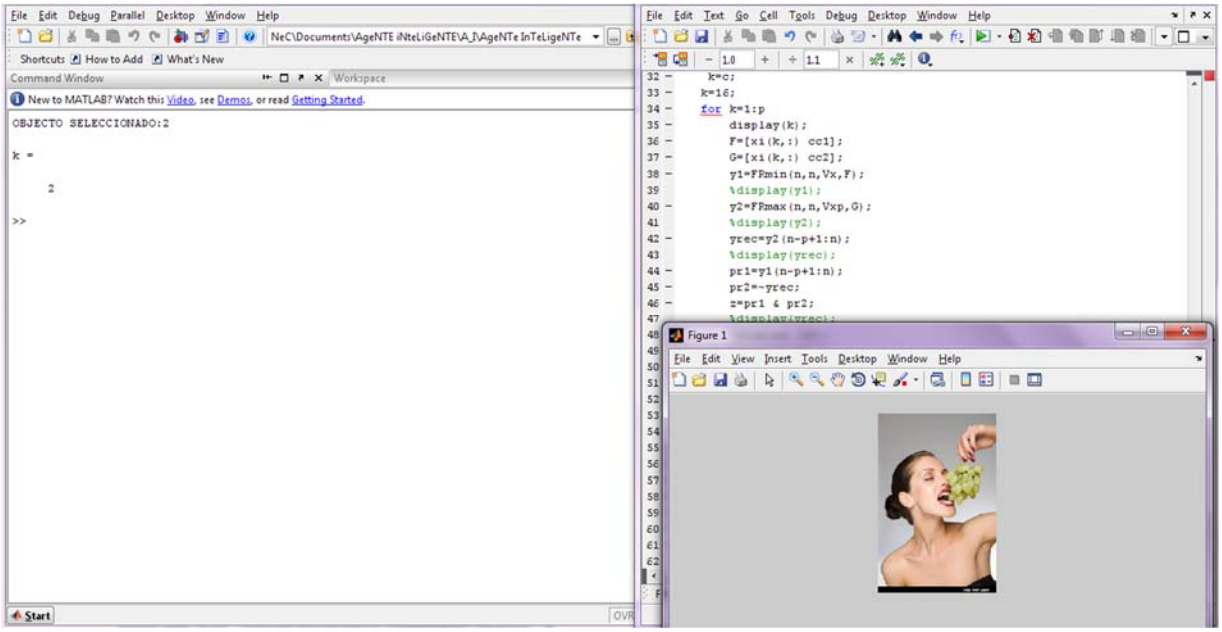


Figura 5.2 Respuesta para el patrón x^2.

Al solicitar se evalué el patrón x^4 , mismo que corresponde a un mago la respuesta asociada será pelar:

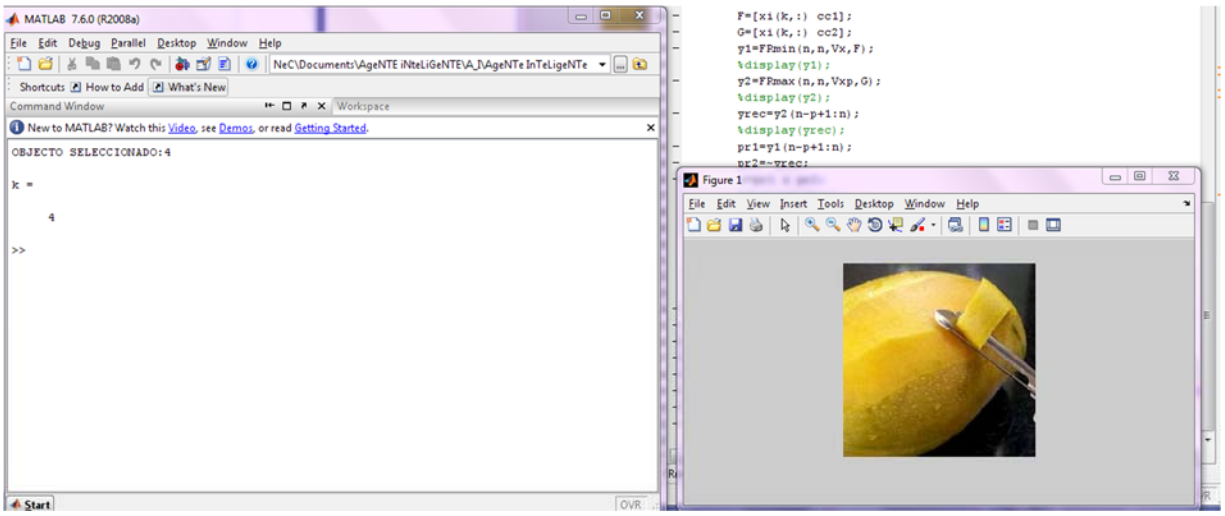


Figura 5.3 Respuesta para el patrón x^4.

Para el patrón x^9, correspondiente al plátano, la acción a ejecutar será pelar:

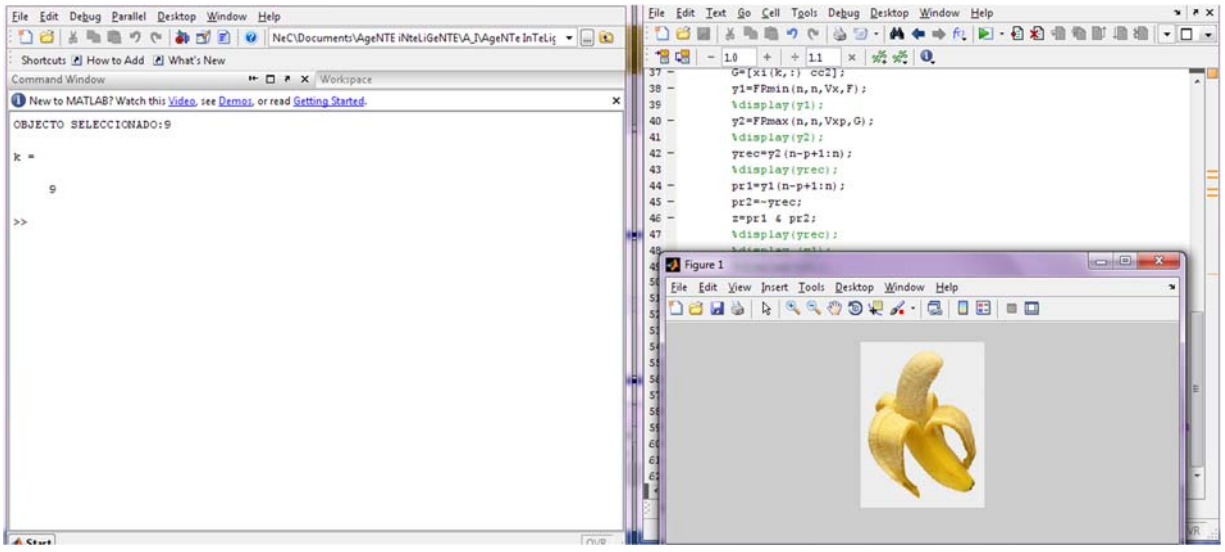


Figura 5.4 Respuesta para el patrón x^9.

Para el patrón x^{12} asociado con un libro de color azul, la acción a ejecutar será abrirlo:

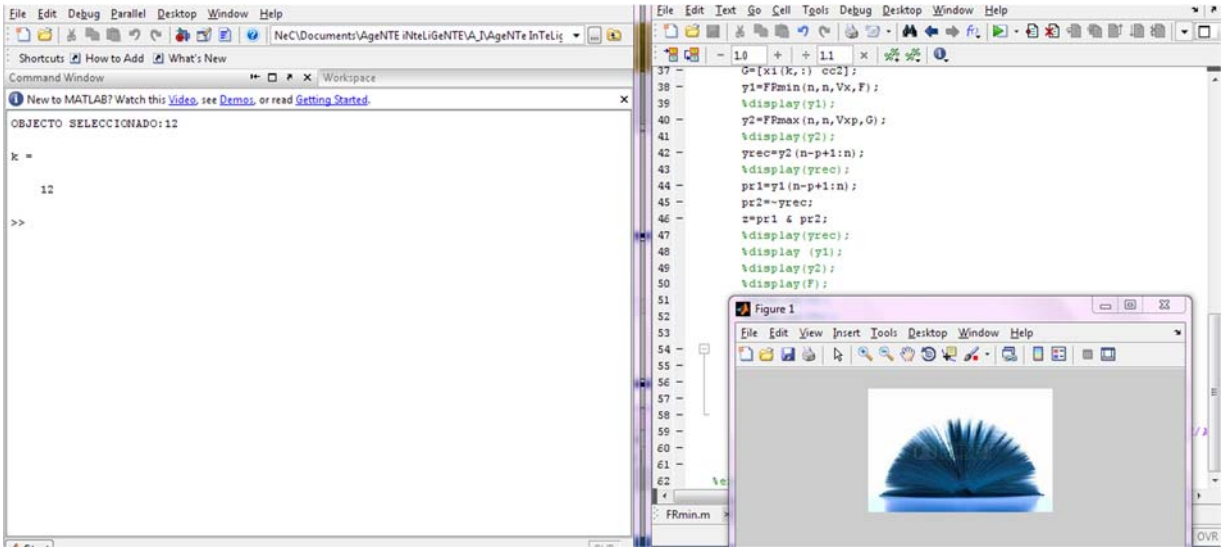


Figura 5.5 Respuesta para el patrón x^{12}.

Para el patrón x^{13}, mismo que corresponde a un ladrillo, se asociara con la acción de arrojar:

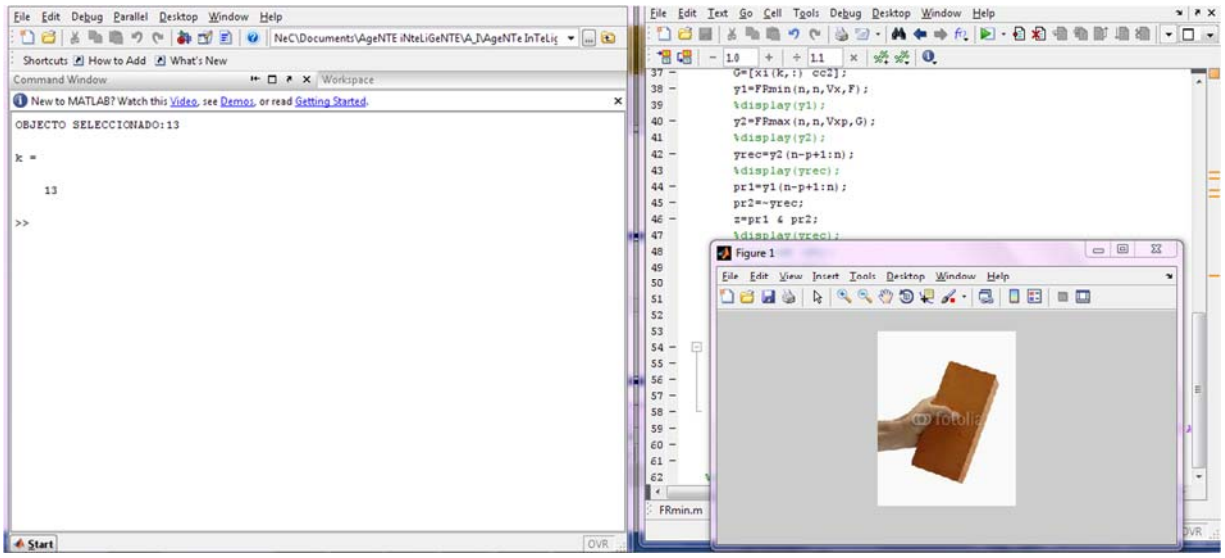


Figura 5.6 Respuesta para el patrón x^{13}.

Para el patrón x^{14} correspondiente a una piedra, la acción correspondiente será al igual que con el ladrillo la de arrojar:

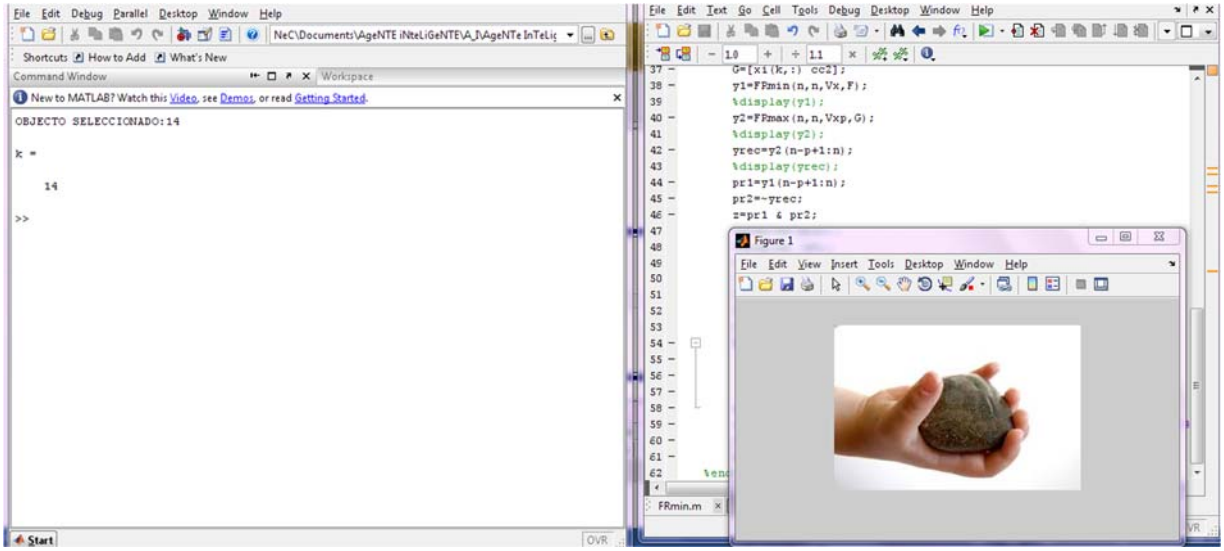


Figura 5.7 Respuesta para el patrón x^{14}.

Para el patrón x^{15} que corresponde a un vaso, la acción a la cual será asociado es a la de beber:

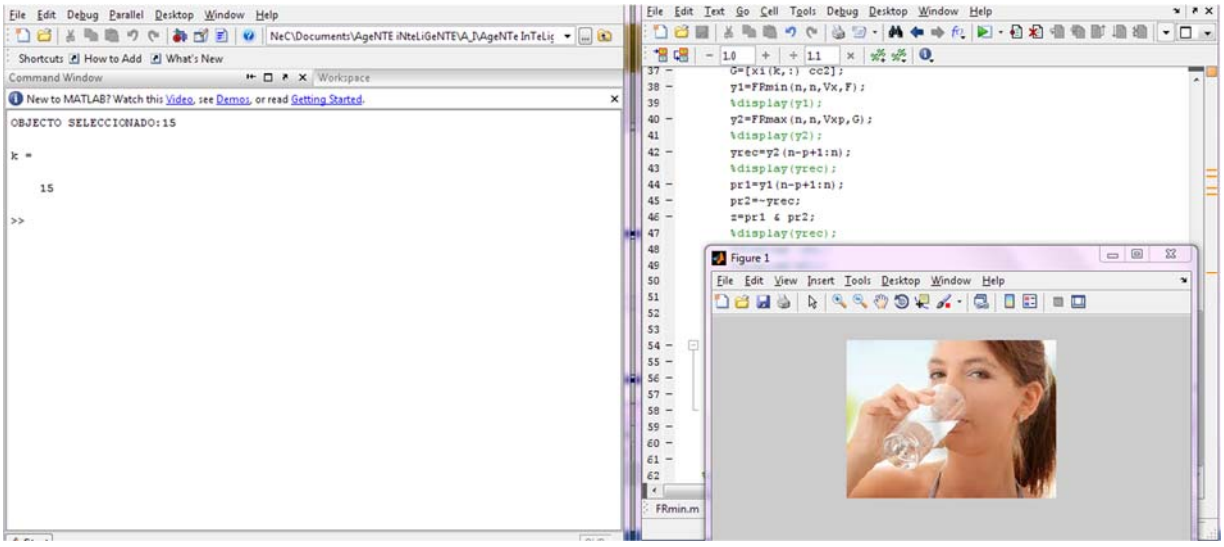


Figura 5.8 Respuesta para el patrón x^{15}.

Finalmente para el patrón x^{16} que es identificado como un lápiz, la acción a ejecutar será la de escribir:

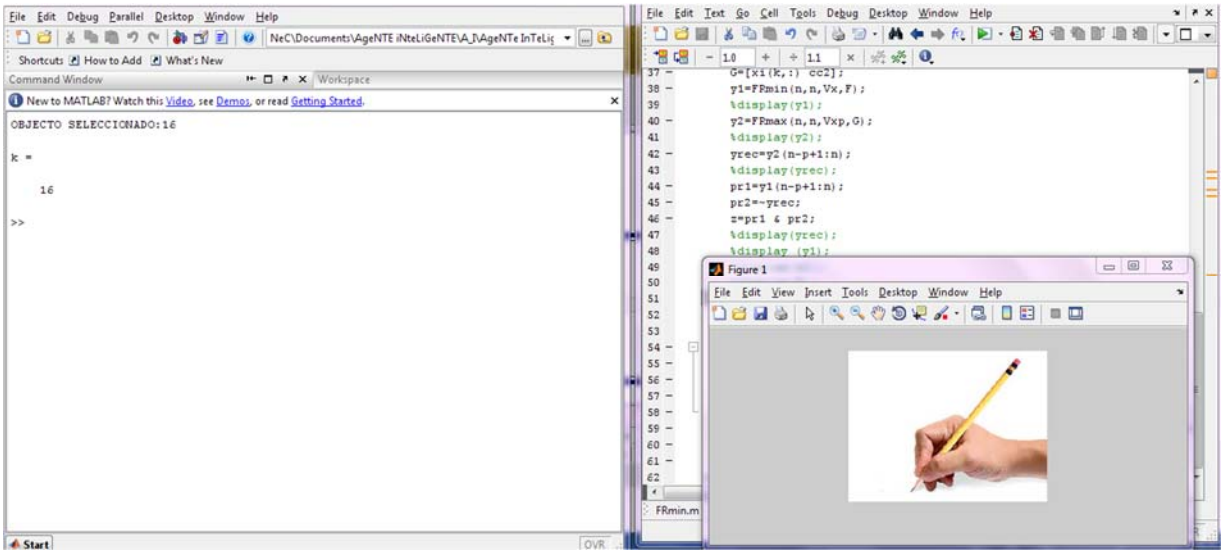


Figura 5.9 Respuesta para el patrón x^{16}.

El mismo procedimiento se realizó para cada una de las 11 frutas y los 5 objetos mencionados anteriormente, mismo que dependiendo su forma y color son clasificados como de tipo fruta o de tipo objeto, para posteriormente relacionarlo con su correspondiente patrón de salida, en este caso ilustrado como la imagen que representa la acción a ejecutar.

Como se puede observar en la tabla 5.1, la respuesta obtenida para cada uno de los patrones de entrada fue positiva, por lo que se cumplió satisfactoriamente con el objetivo planteado.

Tabla 5.1 Resultados obtenidos:

PATRÓN DE ENTRADA	RESULTADO OBTENIDO
x^1 (Uva morada)	Positivo
x^2 (Uva verde)	Positivo
x^3 (Mamey)	Positivo
X^4 (Mango)	Positivo
X^5 (Naranja)	Positivo
X^6 (Guayaba)	Positivo
X^7 (Manzana Roja)	Positivo
X^8 (Manzana verde)	Positivo
X^9 (Plátano)	Positivo
X^{10} (Fresa)	Positivo
X^{11} (Pera)	Positivo
X^{12} (Libro azul)	Positivo
X^{13} (Ladrillo)	Positivo
X^{14} (Piedra)	Positivo
X^{15} (Vaso)	Positivo
X^{16} (Lápiz)	Positivo

SOY POLITÉCNICO

Porque ardo en deseos de despertar al hermano dormido

CONCLUSIONES

Tal y como se planteó en el capítulo 1, el principal objetivo de este proyecto era la implementación de un agente inteligente con Memorias Asociativas Bidireccionales Alfa-Beta, reduciendo el tiempo de aprendizaje que normalmente le tomaría a un agente en aprender por el método RL, o el aprendizaje por imitación.

Al lograr plasmar las características de un objeto (sea comestible o no), en un conjunto de bits, la Alfa-Beta completa satisfactoriamente la fase de aprendizaje y de recuperación para cada uno de los patrones de entrada, por lo que el agente es capaz de asociar, sin error alguno el objeto con la acción que normalmente ejecutaría un ser humano.

Este proyecto se realizó a nivel software, sin embargo, una futura implementación para este agente podría ser a través de reconocimiento de imágenes haciendo uso de las Alfa-Beta, abriendo con ello el camino a una futura implementación a nivel hardware.

SOY POLITÉCNICO

Para prender una antorcha en el altar de la patria

Glosario

Asociativa: Se refiere a relacionar un patrón x^{μ} con un patrón y^{μ}

Autoasociativa: Asociación que sucede cuando el patrón de entrada es igual al de salida.

Bidireccional: Esto sucede cuando al presentarle una entrada **x,** el sistema entrega una salida **y**, de igual forma, si la dirección hacia atrás se lleva a cabo presentándole al sistema una entrada **y** para recibir una salida **x.**

Memoria: La capacidad de poder retener la información que se le presente en la entrada.

Memoria Asociativa: Puede formularse como un sistema de entrada y salida donde el patrón de entrada se representa por un vector **x** y se le asocia el patrón de salida como un vector **y**.

Patrón: es la representación abstracta en unidades de información de un suceso u objeto en el mundo físico.

Recuperación correcta: Se dice que se presenta recuperación perfecta cuando a una memoria asociativa **M** se le presenta un patrón $\mathbf{x}^{w}$ en su entrada y esta responde con el correspondiente patrón fundamental de salida $\mathbf{y}^{w}$.

IA: Inteligencia Artificial, es decir, el arte e ingenio de crear máquinas inteligentes, especialmente programas de cómputo inteligentes.

Agente: Proviene del latín *agere* y se refiere a cualquier entidad capaz de razonar.

Agente Inteligente: Puede ser una entidad fisica o virtual y se define como aquel sistema capaz de percibir su entorno y actuar sobre é

SOY POLITÉCNICO

Porque me dignifico y siento el deber de dignificar a mi institución

REFERENCIAS

[1] McCarthy J,."*The Dartmouth Summer Research Project on Artificial Intelligence*".1956.

[2] Stuart J. Russell, Peter Norvig."Inteligencia artificial: Un enfoque moderno" Ed. Pearson, España Madrid 2004. 2ª ed. Pp 47-49.

[3] Sutton, R., Barto, A.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (1998).

[4] Wiewiora, E., Cottrell, G., Elkan, C.: Principled methods for advising reinforcement learning agents. In: Proceedings of the Twentieth International Conference on Machine Learning (ICML 2003), pp. 792–799 (2003).

[5] Billard, A., Epars, Y., Calinon, S., Schaal, S., Cheng, G.: Discovering optimal Imitation strategies. Robotics and Autonomous Systems 47(2-3), 69–77 (2004).

[6] Alissandrakis, A., Nehaniv, C., Dautenhahn, K.: Imitation with Alice: Learning To imitate corresponding actions across dissimilar embodiments. IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans 32(4), 482–496 (2002).

[7] Sakato, T., Ozeki, M., Oka, N.: A Computational Model of Imitation and Autonomous Behavior in Continuous Spaces. R. Lee (Ed), Networking and Parallel Distributed Computing (SNPD),SCI 492, pp. 37–51(2013).

[8] Acevedo, E. 2006. Memorias asociativas bidireccionales Alfa-Beta, Tesis de Doctorado. Centro de Investigación en Computación, México.

[9] Simpson, P.K.1990, *Artificial Neural Systems*, Pergamon Press, New York.

[10] Minsky, M.& Papert,S. 1969, MIT Press,Cambridge.

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

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

[13] Willshaw, D., Buneman, O. & Longuet-Higgins, H. 1969, "Non-holograpic associative memory", Nature, no. 222, pp. 960-962.

[14] Anderson, J. A.1972,"A simple neural network generating an Interactive memory", Mathematical Biosciencies, vol.14, pp. 197-220.

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

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

[17] Sossa, H., Barrón, R. & Vázquez, R. 2004, "New Associative Memories to Recall Real-Valued Patterns", CIARP, LNCS 3287, pp. 195-202.