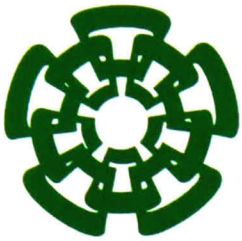
CT-723-551
Lon.-2013

xx(202805.1)



Centro de Investigación y de Estudios Avanzados
del Instituto Politécnico Nacional
Unidad Guadalajara

Redes neuronales geométricas multicapa y retroalimentadas

CINVESTAV
IPN
ADQUISICION
LIBROS

Tesis que presenta:

Eduardo Filemón Vázquez Santacruz

para obtener el grado de:

Doctor en Ciencias

en la especialidad de:

Ingeniería Eléctrica

Director de Tesis

Dr. Eduardo José Bayro Corrochano

CINVESTAV del IPN Unidad Guadalajara, Guadalajara, Jalisco, Junio de 2012.

CLASSIF.: CT00627
ACCESSION: CT-723-SSI
FILMED: 15-03-2013
FILMED BY: Dan: 2013
ID: 202622-1001

Centro de Investigación y de Estudios Avanzados
del Instituto Politécnico Nacional

Unidad Guadalajara

Departamento de Ingeniería Eléctrica y Ciencias de la Computación

Redes neuronales geométricas multicapa y retroalimentadas

Tesis que presenta

Eduardo Filemón Vázquez Santacruz

Director de la Tesis

Dr. Eduardo José Bayro Corrochano

Guadalajara, Jalisco. **4 de junio de 2012**

Resumen

Este trabajo de investigación se ha desarrollado en el marco de diseño y análisis de algoritmos, específicamente de redes neuronales artificiales utilizando el marco matemático de las álgebras geométricas de Clifford y considerando un conjunto de datos formulados mediante técnicas geométricas, entonces es necesario desarrollar algoritmos que procesen este tipo de datos. El desarrollo de algoritmos específicos para este tipo de información nos permite un mejor entendimiento y un mejor desempeño del procesamiento.

La redes neuronales biológicas muestran una variedad de diferentes maneras de procesar información. Los algoritmos que hemos desarrollado son capaces de desarrollar un procesamiento de información geométrica de manera que internamente también consideran parámetros geométricos que describen de manera más clara diversos aspectos relacionados con el procesamiento de la información de entrada.

El álgebra geométrica es el marco de trabajo matemático que ofrece diversas herramientas para realizar este tipo de análisis de manera más simple. Los objetos externos a los algoritmos pueden ser descritos de una manera más clara y la interpretación de las manipulaciones geométricas de diversas entidades gemétricas se hace más fácil.

En esta tesis, se desarrollaron redes geométricas RBF recurrentes y de tipo *feed-forward*. Estas redes geométricas se utilizaron para abordar ciertos problemas de visión robótica. Estas redes no tienen paralelo en la literatura y son nuevas porque con ellas se obtiene de forma explícita los mapeos involucrados. Los mismos resultados no se puede lograr utilizando redes neuronales RBF valuadas con reales.

Glosario

Símbolo	Descripción
AG	Álgebra Geométrica
CGA	Álgebra Geométrica Conformal
$\underline{x}$	Punto Conformal
$\mathbf{x}$	Punto Euclidiano
$\mathbf{e}_i$	Base e_i
$\mathbf{I}$	Pseudo-Escalar unitario
$\mathbf{E}$	Plano de Minkowski
$\underline{PP}$	Par de puntos en CGA
$\underline{s}$	Esfera en CGA
$\underline{z}$	Círculo en CGA
$\underline{\pi}$	Plano en CGA
$\underline{L}$	Línea en CGA
$\underline{R}_\theta$	Rotor con ángulo θ en CGA
$\underline{A}^*$	Representación dual del multivector $\underline{A}$
$\widetilde{\underline{A}}$	Reversión del multivector $\underline{A}$
$\widehat{\underline{A}}$	Involución principal del multivector $\underline{A}$
$\mathbf{M}$	Motor en el álgebra $\mathcal{G}_{p,q,r}$
$\mathbf{R}$	Rotor en el álgebra $\mathcal{G}_{p,q,r}$
$\mathbf{T}$	Translador en el álgebra $\mathcal{G}_{p,q,r}$

Abstract

This research work has been developed within the context of design and analysis of algorithms, specifically artificial neural networks using the mathematical framework of Clifford geometric algebras and considering a data set made using geometric techniques, then it is needed to develop algorithms to process this type of data. The development of specific algorithms for this type of information allows us to better understanding and better processing performance.

The biological neural networks show a variety of different ways of processing information. The algorithms we have developed are able to develop a geometric processing information internally so that they also consider geometric parameters which more clearly describe various aspects of processing the input information.

The geometric algebra is the mathematical framework that provides tools for this type of analysis in a more simple. Objects outside of the algorithms can be described in a more clear and the interpretation of geometric manipulations of various entities gemetricas becomes easier.

In this thesis, feed-forward and recurrent RBF geometric networks were developed. These geometric networks were used to tackle certain problems in robot vision. These networks have no parallel in the literature and are novel because they yield explicitly the involved mapping. Same results can not achieved using real valued RBF networks.

In this work we use vision techniques for preprocessing of the input data to the algorithms. With the help of a stereo vision system, we represent some three-dimensional objects that belong to the external environment. These are

Agradecimientos

Al Consejo Nacional de Ciencia y Tecnología por los recursos otorgados que facilitaron los estudios de doctorado. Al personal del Departamento de Ingeniería Eléctrica y Ciencias de la Computación del Centro de Investigación y de Estudios Avanzados del IPN, Unidad Guadalajara, por los conocimientos y hospitalidad ofrecidos. Al personal de la Biblioteca de la Unidad Guadalajara por facilitar el material necesario durante los estudios del doctorado.

Al Prof. Dr. Eduardo Bayro Corrochano, mi asesor en el doctorado, por su singular motivación, asesoría y enseñanza durante el trabajo de tesis realizado.

A mi asesor, en la maestría, el Dr. Debrup Chakraborty, por su incondicional apoyo y motivación.

A los Drs. Mario Siller González, Alexander Loukianov, Nancy Arana Daniel y Sussana Ortega por sus valiosos comentarios durante la revisión del documento de tesis.

En especial a mis padres, quienes han fomentado parte de la integridad que me ha forjado como un ser humano comprometido con su alrededor.

Gracias a la vida que permite la posibilidad de dudar, pensar, imaginar y crear.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

INTRODUCCIÓN

Un procesamiento eficaz de la información en el mundo real debe ser lo suficientemente flexible para enfrentar un ambiente que cambia inesperadamente y dinámicamente. Como consecuencia, tal procesamiento es considerado inseparable de la existencia física de los fenómenos cambiantes en el mundo. Por lo tanto, la inteligencia artificial experimenta un cambio de paradigma en estas décadas para introducir un marco de trabajo de procesamiento no simbólico a través de las Redes Neuronales Artificiales (RNAs). Un punto fuerte de las RNAs radica en la adaptabilidad asi como en las habilidades de aprendizaje y de auto-organización. El procesamiento en las RNAs es largamente influenciado por la acumulación de experiencia (lo que ellas han percibido y obtenido desde el ambiente). El cerebro tiene una muy alta adaptabilidad a las circunstancias físicas que el medio ambiente determina y está vinculado con órganos de sentido tales como ojos, oídos, nariz, sensores cutáneos y otros, para sensar el ambiente, asi como también con órganos motores para trabajar en el ambiente tales como manos, piernas, cuerdas vocales, músculos en general y otros. La idea básica más importante en las RNAs valuadas con números Hiper-complejos (RNVHs) es extender las propiedades de procesamiento fundamentales del cerebro [3].

Dado que las funciones neuronales (aprendizaje y autoorganización) son influenciadas por interfaces sensoriomotoras que conectan la red neuronal con el ambiente, existen ciertas situaciones donde las RNVH son inevitablemente

requeridas o muy efectivas [3, 1]

1.1. Motivación

Los estudios sobre redes neuronales valuadas con hiper-complejos han estado evolucionando recientemente en varias direcciones. Se han creado redes neuronales valuadas con números complejos (RNVCs) [3] y las áreas pioneras de su aplicación incluyen detección de ondas electromagnéticas y luminosas e imágenes y restauración en procesamiento de imágenes. Las RNVHs pueden usarse en el estudio de procesamiento adaptivo de señales para el control en ambientes cambiantes y desconocidos, procesamiento de información simulando el del cerebro, robótica inspirada por sistemas neuronales humanos y más. Las RNVHs tratan con información valuada con hiper-complejos usando parámetros y variables valuadas con hiper-complejos. Es significantemente importante para la red neuronal humana adoptar una representación adecuada para el propósito asignado a módulos respectivos de la red. Por lo tanto, la red de cada módulo también posee una construcción adecuada para el procesamiento de información respectiva específica para señales visuales, auditivas u olfativas. La electrónica moderna y comunicaciones nos proporcionan una larga variedad de información. Por lo tanto, en los campos de la ingeniería se espera desarrollar nuevos sistemas que procesen una gama más amplia de información de manera más efectiva y adaptativa, como nosotros los humanos lo hacemos, o mejor. En tales casos, el aprendizaje y auto organización requiere de representaciones de información adecuadas para tales propósitos. La idea es crear un tipo de cerebro artificial enriqueciendo la representación de la información. Una de las ventajas más importantes de las RNVHs es la buena compatibilidad con fenómenos de onda, como es el caso para las RNVCs. Existen fenómenos que son expresados mas simplemente y naturalmente mediante números hiper-complejos. Éstos también están relacionados directamente con los procesos elementales en las RNVHs tales como la ponderación en las conexiones sinápticas y la sumatoria de las entradas ponderadas. La manera básica de procesamiento de naturaleza biológica es común tanto a las redes neuronales convencionales (valuadas con números reales) como a las

RVNHs.

El mundo externo, parece ser comprendido por el ser humano en términos de representaciones geométricas intrínsecas. Es posible formalizar las relaciones entre las señales físicas del mundo externo y las señales internas del ser humano usando vectores extrínsecos para representar aquellas señales que llegan del mundo y vectores intrínsecos para representar las señales del mundo interior [4, 5].

1.2. El problema a resolver

Se asume que los mundos interno y externo emplean diferentes sistemas de coordenadas de referencia. Al considerar a la adquisición y codificación del conocimiento un proceso distruibuido y diferenciado, entonces se puede pensar que debe existir varios dominios de representación del conocimiento que obedecen diferentes métricas y que pueden ser modelados usando bases vectoriales diferentes. Este tipo de problemas considera representaciones para tratar con procesamiento de señales [6]. La formalización de la representación geométrica parece ser un proceso dual que implica la expresión de señales físicas extrínsecas construidas por vectores intrínsecos del sistema nervioso central [4] [5]. Estas representaciones vectoriales, relacionadas a marcos de referencia intrínsecos al ser humano, se relacionan al análisis de percepción y síntesis de acciones. El mapeo geométrico entre estos dos espacios vectoriales puede por lo tanto ser implementado por una red neuronal que funcione como un tensor métrico [5].

En este sentido, se puede usar el álgebra geométrica (AG) para ofrecer una alternativa al análisis tensorial que ha sido empleado desde 1980 por Pellionisz A. y Llinàs para la teoría del ciclo percepción acción (CPA). Nuestro trabajo se enfoca en el diseño de modelos neuronales que puedan tratar con información geométrica externa e interna para tener una descripción más natural e intuitiva de la implementación y sus resultados.

El cálculo tensorial requiere reglas de transformación para definir relaciones independientes de coordenadas. El AG es más atractiva que el análisis

tensorial porque es libre de coordenadas (un álgebra de direcciones) e incluye *spinors*, lo que no permite la teoría de tensores. La eficiencia computacional del AG ha sido confirmada en muchas áreas difíciles de física-matemática [7] como cálculo geométrico, y tiene profunda relevancia en física cuántica. También tiene aplicaciones en robótica, visión computacional, procesamiento de imágenes, procesamiento de señales y dinámica espacial. El análisis matricial es otro sistema matemático usado para formular redes neuronales. Además, el AG ofrece otras ventajas computacionales que el álgebra de matrices no, como la representación con "bivectores" de operadores lineales, relaciones de incidencia (operaciones *meet* y *join*), y el grupo conformal en la horósfera. Los intentos iniciales para aplicar el AG a la geometría neuronal se describen en [8, 9, 10, 11]. Posteriormente se desarrollaron otros modelos como [12, 13, 14, 15, 16, 17, 18]. En [19] se da un tutorial completo de este tipo de modelos. Hoy se sabe que las redes estándares de "alimentación hacia adelante" son generalizables usando AG [1]. En cuanto a las redes neuronales recurrentes simples, prácticamente no existe trabajo alguno donde se considere el lenguaje geométrico para expresar los parámetros de la arquitectura neuronal [20].

El AG ofrece una interpretación geométrica natural de los problemas a resolver, lo que permite una mayor intuición que el análisis vectorial estándar. Es más eficiente porque reduce el número de operaciones a considerar. Está bien definida para dimensiones superiores e inferiores. Ofrece una alternativa práctica a los métodos convencionales para manipular con facilidad entidades geométricas 3D (puntos, líneas, planos) con diversas operaciones como son: reflexiones y rotaciones.

Las RNVCs han sido estudiadas para los casos de una única neurona. perceptrones organizados en múltiples capas y redes recurrentes [16][3]. Estas redes trabajan con información, parámetros y variables valuadas con complejos. Las entidades valuadas con hiper-complejos, es decir, un sistema de números con dimensión mayor que el valor complejo, también son usadas para modelar redes neuronales. Este tipo de entidades puede ser encontrado en un sistema algebraico como el álgebra de cuaterniones. En este sistema matemático, los investigadores han propuesto neuronas y redes como la neurona cuaterniónica y los MLPs cuaterniónicos [3, 17, 12, 15, 1].

Algunos investigadores que han trabajado con RNVCs, han desarrollado redes valuadas con hiper-complejos usando entidades geométricas [1]. Li, et al. En [16] se propuso la red neuronal de funciones de base radial (RBF-N) valuada con complejos. Sin embargo, no existe aún trabajos para el desarrollo de RBF-Ns valuadas con hiper-complejos de mayor dimensión.

En este trabajo, se proponen modelos de redes neuronales artificiales multicapa y retroalimentadas donde las entradas, las salidas, pesos y neuronas ocultas son expresados usando multivectores, con la gran diferencia de que se pueda extraer de las capas ocultas información geométrica pertinente a las transformaciones que sufren los datos de entrada.

1.3. Objetivos

El AG permite operar con objetos geométricos de una manera natural. En este marco de trabajo desarrollamos nuevos modelos de redes neuronales que son capaces de tratar con objetos definidos geométricamente con una arquitectura y proceso de aprendizaje también definidos en términos geométricos.

Los objetivos son:

- Desarrollar una red neuronal geométrica estática.
- Desarrollar una red neuronal geométrica recurrente.

1.4. Organización del documento

El capítulo dos describe el AG y su uso para realizar diversos cálculos, se describen sus entidades y reglas para trabajar con ella; el uso del AG de motores con sus entidades y sus representaciones en el espacio, así como las propiedades de algunas de las entidades del álgebra. El capítulo tres explica los conceptos necesarios para entender la importancia de las redes neuronales extendidas con valores multivectoriales. El capítulo cuatro describe los algoritmos desarrollados y su aplicación usando el AG. El capítulo cinco muestra un

análisis experimental de nuestro trabajo y finalmente el capítulo seis expresa las conclusiones y trabajo futuro de nuestra investigación.

Capítulo 2

COMPUTACIÓN GEOMÉTRICA

La geometría sintética, geometría de coordenadas, números complejos, cuaterniones, análisis vectorial y de tensores, álgebra de matrices, álgebra de Grassmann, álgebra de Clifford, álgebra de *Spinors*, cinemática, espacio de Plücker y geometría proyectiva poseen conceptos geométricos. Además, existen consecuencias innecesarias de muchos lenguajes como el aprendizaje redundante, complejidad de acceso al conocimiento, traducción frecuente, y más. Esto es una buena razón para considerar un lenguaje que involucre las técnicas anteriores para resolver problemas relacionados..

El álgebra geométrica (AG) es un lenguaje para geometría que explota el concepto de un vector. Define "multivectores" mediante la combinación de representaciones con dimensiones diferentes: escalares, vectores, bi-vectores, trivectores y k-vectores. Usa dimensiones de las representaciones llamadas *grades*. Es adecuada para representar estructuras y desarrollar algoritmos que se apliquen en ingeniería. Ofrece las herramientas necesarias y de unificación para expresar geometría y sus relaciones sin la necesidad de cambiar de sistema matemático o hacer excepciones en casos especiales, esto permite que la comunicación de las ideas sea más fácil, lo que da lugar a desarrollar algoritmos de manera rápida e intuitiva [21]. Ofrece el potencial para realizar optimizaciones e implementaciones altamente eficientes y se confía que será competitiva con métodos clásicos cuando también se adapten algoritmos a sus nuevas capacidades [22].

Tales técnicas de ahorro de cálculos podrían fácilmente compensar la ligera pérdida de velocidad por el cálculo con representaciones en dimensiones más altas. Estos algoritmos deben ser desarrollados si el AG es aplicada en el mundo real. Se debe considerar que en un amplio rango de aplicaciones de ingeniería muchos sistemas matemáticos diferentes están concurrentemente en uso y que las implementaciones sencillas se ejecutan con lentitud por el tamaño del álgebra [23]. El AG como lenguaje matemático frecuentemente sugiere una estructura más clara y de mayor elegancia en el entendimiento de métodos y fórmulas. Esto permite mayor eficiencia y desempeño con tiempo de ejecución menor para los algoritmos derivados [22, 24].

2.1. Introducción al álgebra geométrica

El AG ofrece una alternativa y un enfoque comprensivo a la representación algebraica de la geometría de varias áreas de investigación [1, 25, 23, 26, 24, 25]. Se aplica además en física, gráficación y robótica [24]. Una característica distintiva del AG es que sus productos son usados e interpretados geométricamente debido a la correspondencia entre entidades geométricas y elementos del álgebra. Permite que uno manipule subespacios directamente y es un formalismo libre de coordenadas. También introduce un concepto más general que el producto cruz el cuál es únicamente definido en tres dimensiones. Proporciona descripciones compactas e intuitivas en sus áreas de aplicación. Comparada con el álgebra lineal, el rico lenguaje matemático del AG conduce, después de elegir el álgebra, a menos trabajo para implementar la aplicación.

2.1.1. El producto Clifford de vectores en $\mathbb{R}^2$

Es importante tener una multiplicación de vectores que satisfaga los axiomas de multiplicación de los números reales - distributividad, asociatividad y conmutatividad - y que la norma de estos vectores se preserve en dicha multiplicación $|\mathbf{ab}| = |\mathbf{a}||\mathbf{b}|$, donde $\mathbf{a}$ y $\mathbf{b}$ son vectores. A esta multiplicación de vectores la llamamos "producto Clifford". Dado que es imposible man-

tener una conmutabilidad en dimensiones de $n \geq 3$, nos enfocaremos en la distribución y asociación.

Consideremos dos vectores con base ortonormal $\mathbf{e}_1 = (1,0)^T$ y $\mathbf{e}_2 = (0,1)^T$ en el espacio vectorial $\mathbb{R}^2$. La magnitud del vector $\mathbf{r} = x\mathbf{e}_1 + y\mathbf{e}_2$ es $|\mathbf{r}| = \sqrt{x^2 + y^2}$. Si el vector $\mathbf{r}$ es multiplicado utilizando el producto Clifford con él mismo, la expresión resultaría en $\mathbf{rr} = \mathbf{r}^2$, que es igual al cuadrado de su magnitud.

$$\mathbf{r}^2 = |\mathbf{r}|^2 \tag{2.1}$$

Usando la forma de coordenadas, introduciremos el producto de vectores de tal forma que

$$(x\mathbf{e}_1 + y\mathbf{e}_2)^2 = x^2 + y^2 \tag{2.2}$$

Utilizando la regla distributiva sin asumir la conmutabilidad obtenemos

$$x^2\mathbf{e}_1{}^2 + y^2\mathbf{e}_2{}^2 + xy(\mathbf{e}_1\mathbf{e}_2 + \mathbf{e}_2\mathbf{e}_1) = x^2 + y^2 \tag{2.3}$$

y entonces se puede definir el producto de vectores (el producto Clifford).

Definición 2.1 *Sean* $\mathbf{a}, \mathbf{b} \in \mathbb{R}^2$, *dos vectores con una base ortonormal* $\mathbf{e}_1 = (1,0)^T$ *y* $\mathbf{e}_2 = (0,1)^T$, *el producto Clifford de los vectores* $\mathbf{a} = a_1\mathbf{e}_1 + a_2\mathbf{e}_2$ *y* $\mathbf{b} = b_1\mathbf{e}_1 + b_2\mathbf{e}_2$ *está dado por:*

$$\mathbf{ab} = a_1b_1 + a_2b_2 + (a_1b_2 - a_2b_1)\mathbf{e}_1\mathbf{e}_2 \tag{2.4}$$

Esto se satisface si los vectores ortogonales $\mathbf{e}_1, \mathbf{e}_2$ obedecen las reglas de multiplicación

$$\begin{matrix} \mathbf{e}_1{}^2 = \mathbf{e}_2 = 1 \\ \mathbf{e}_1\mathbf{e}_2 = -\mathbf{e}_2\mathbf{e}_1 \end{matrix} \quad \text{que corresponde con} \quad \begin{matrix} |\mathbf{e}_1| = |\mathbf{e}_2| = 1 \\ \mathbf{e}_1 \perp \mathbf{e}_2 \end{matrix} \tag{2.5}$$

Se usa la propiedad asociativa para calcular el cuadrado $(\mathbf{e}_1\mathbf{e}_2)^2 = (\mathbf{e}_1\mathbf{e}_2)(\mathbf{e}_1\mathbf{e}_2) = \mathbf{e}_1(\mathbf{e}_2\mathbf{e}_1)\mathbf{e}_2 = -\mathbf{e}_1{}^2\mathbf{e}_2{}^2 = -1$. Dado que el cuadrado del producto $\mathbf{e}_1\mathbf{e}_2$ es negativo, se sigue que $\mathbf{e}_1\mathbf{e}_2$ no es un escalar ni un vector. El producto es una nueva clase de unidad llamado **bivector**, representando el área del plano orientado (Figura 2.1) del cuadrado con los lados $\mathbf{e}_1$ y $\mathbf{e}_2$. Se escribe $\mathbf{e}_{12} = \mathbf{e}_1\mathbf{e}_2$.

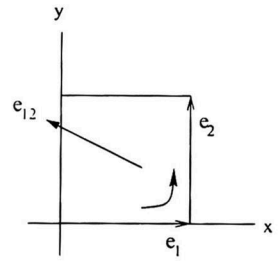


Figura 2.1: Bivector; el plano orientado

2.1.2. Producto exterior (parte bivectorial del producto Clifford)

Extrayendo las partes escalar[1] y bivector del producto Clifford, tenemos como producto de dos vectores $\mathbf{a} = a_1\mathbf{e}_1 + a_2\mathbf{e}_2$ y $\mathbf{b} = b_1\mathbf{e}_1 + b_2\mathbf{e}_2$

$$\mathbf{a} \cdot \mathbf{b} = a_1b_1 + a_2b_2. \quad \text{El producto escalar "a punto b",} \tag{2.6}$$

$$\mathbf{a} \wedge \mathbf{b} = (a_1b_2 - a_2b_1)\mathbf{e}_{12}. \quad \text{El producto exterior "a wedge b",} \tag{2.7}$$

El bivector $\mathbf{a} \wedge \mathbf{b}$ representa el segmento del plano orientado del paralelogramo con lados **a** y **b**. El área de este paralelogramo (Figura 2.2) es $|a_1b_2 - a_2b_1|$, y tomaremos la magnitud del bivector $\mathbf{a} \wedge \mathbf{b}$ para determinar el área $|\mathbf{a} \wedge \mathbf{b}| = |a_1b_2 - a_2b_1|$.

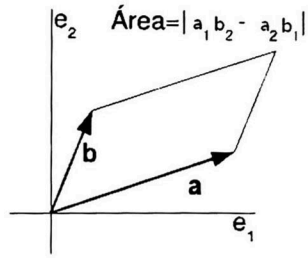


Figura 2.2: Área de un paralelogramo

El paralelogramo puede ser visto como una clase de producto geométrico

[1] El producto escalar también es conocido como **producto punto** o **producto interior**

de sus lados

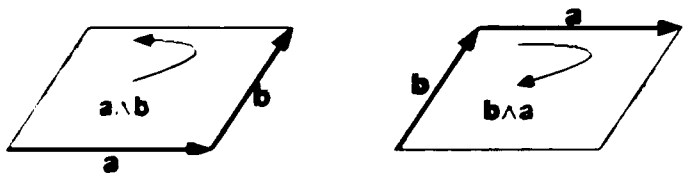


Figura 2.3: Sentido del bivector

El bivector $\mathbf{a} \wedge \mathbf{b}$ y $\mathbf{b} \wedge \mathbf{a}$ tienen la misma magnitud pero con sentidos de rotación opuestos (Figura 2.3). Esto puede ser expresado simplemente escribiendo

$$\mathbf{a} \wedge \mathbf{b} = -\mathbf{b} \wedge \mathbf{a} \tag{2.8}$$

Se observa que el producto Clifford de dos vectores

$$(a_1\mathbf{e}_1 + a_2\mathbf{e}_2)(b_1\mathbf{e}_1 + b_2\mathbf{e}_2) = a_1b_1 + a_2b_2 + (a_1b_2 - a_2b_1)\mathbf{e}_{12} \tag{2.9}$$

es la suma de su parte escalar $(a_1b_1 + a_2b_2)$ y un bivector $(a_1b_2 - a_2b_1)$.

El producto Clifford así como el producto exterior y producto interior (o escalar), puede ser llevado a mayores dimensiones (ejemplo $\mathbb{R}^n$, donde $n \geq 2$).

2.2. Álgebra geométrica

El **álgebra geométrica** $\mathcal{G}_n$ es un espacio vectorial *n-dimensional* que puede ser expandido usando las bases ortonormales de sus vectores $\{\mathbf{e}_i\}, i = 1, \ldots, n$. tal que $\mathbf{e}_i\mathbf{e}_j = \mathbf{e}_{ij}$.

$$1, \{\mathbf{e}_i\}, \{\mathbf{e}_i\mathbf{e}_j\}, \{\mathbf{e}_i\mathbf{e}_j\mathbf{e}_k\}, \cdots, \mathrm{I} \equiv \mathbf{e}_1\mathbf{e}_2\cdots\mathbf{e}_n \tag{2.10}$$

Note, para $i \neq j$, se asume $e_ie_j = e_i \cdot e_i + e_i \wedge e_j = e_i \wedge e_j$.

En general, una álgebra geométrica $\mathcal{G}_{p,q,r}$ $(p, q, r, \in \mathbb{N})$ es un espacio lineal de dimensión 2^n, $n = p + q + r$, con un subespacio expandido por k-vectores o *blades*, en donde p,q y r corresponden al número de vectores base que cuadran

a +1, -1 y 0, respectivamente. La notación $\mathcal{G}_{p,q,r}(\mathbb{R}^{p,q,r})$ es usada para determinar el espacio vectorial $\mathbb{R}_{p,q,r}$ equivalente de donde provienen los elementos del álgebra.

Sean $\mathbf{e}_i$ y $\mathbf{e}_j$ ($\mathbf{e}_i, \mathbf{e}_j \in \mathbb{R}^{p,q,r}$) dos vectores de la base ortonormal de un espacio vectorial $\mathcal{G}_{p,q,r}$. Entonces el producto geométrico de éstos nos da la base del AG $\mathcal{G}_{p,q,r}$ y está definida como

$$\mathbf{e}_i\mathbf{e}_j := \begin{cases} 1 \in \mathbb{R} & \text{para } i = j \in \{1, \ldots, p\} \\ -1 \in \mathbb{R} & \text{para } i = j \in \{p+1, \ldots, p+q\} \\ 0 \in \mathbb{R} & \text{para } i = j \in \{p+q+1, \ldots, n\} \\ \mathbf{e}_{ij} = \mathbf{e}_i \wedge \mathbf{e}_j = -\mathbf{e}_j \wedge \mathbf{e}_i & \text{para } i \neq j. \end{cases} \tag{2.11}$$

El producto geométrico (o Clifford) de dos vectores se expresa como

$$\mathbf{ab} = \mathbf{a} \cdot \mathbf{b} + \mathbf{a} \wedge \mathbf{b} \tag{2.12}$$

La regla conmutativa $\mathbf{a}\cdot\mathbf{b} = \mathbf{b}\cdot\mathbf{a}$ junto con la regla anticonmutativa $\mathbf{a}\wedge\mathbf{b} = -\mathbf{b}\wedge\mathbf{a}$ nos da la relación que hay entre **ab** y **ba**. Así

$$\mathbf{ba} = \mathbf{a} \cdot \mathbf{b} - \mathbf{a} \wedge \mathbf{b} \tag{2.13}$$

Sumando y restando las ecuaciones (2.12) y (2.13), podemos definir

$$\mathbf{a} \cdot \mathbf{b} = \frac{1}{2}(\mathbf{ab} + \mathbf{ba}) \tag{2.14}$$

$$\mathbf{a} \wedge \mathbf{b} = \frac{1}{2}(\mathbf{ab} - \mathbf{ba}) \tag{2.15}$$

La parte anti-simétrica del producto geométrico es definida por el producto exterior de un r-vector. Este resultado es llamado *r-blade* o *blade de grado r*. Una combinación lineal de los *r-blades* es llamado *r-vector*. El conjunto de *r-vectores* es un subespacio $\binom{n}{r}$-dimensional de $\mathcal{G}_n$, denotado por $\mathcal{G}_n^r$. Todo el espacio $\mathcal{G}_n$ está dado por la suma de sus subespacios

$$\mathcal{G}_n = \sum_{i=0}^{n} \mathcal{G}_n^i \tag{2.16}$$

Un elemento genérico de $\mathcal{G}_n$ es llamado "multivector" De acuerdo con (2.16), cada multivector M puede ser expandido como

$$M = \sum_{i=0}^{n} \langle M \rangle_i \tag{2.17}$$

donde $\langle M \rangle_i$ denota la parte *i-vector*.

2.2.1. Propiedades de los multivectores

Un elemento M de $\mathcal{G}_n$ es "invertible" si existe otro elemento N de $\mathcal{G}_n$ tal que $MN = NM = 1$. El elemento N, si existe, es único. Éste es llamado el "inverso" de M, y es denotado por M^{-1}. Por ejemplo, un vector nulo[2] de $\mathcal{G}_n$ no es invertible, pero cualquier vector no nulo **a** es invertible, con

$$\mathbf{a}^{-1} = \frac{1}{\mathbf{a}} = \frac{\mathbf{a}}{\mathbf{a}^2} \tag{2.18}$$

La división de vectores en AG facilita varios cálculos.

Utilizando la asociatividad y multi-linealidad, el producto exterior puede ser aplicado a cualquier número finito de multivectores y a escalares de la manera siguiente

$$\lambda \wedge M = M \wedge \lambda = \lambda M, \text{ para } \lambda \in \mathbb{R}, M \in \mathcal{G}_n \tag{2.19}$$

El producto interior de un *r-blade* $a_1 \wedge \cdots \wedge a_r$ con un *s-blade* $b_1 \wedge \cdots \wedge b_s$ está definido como

$$(a_1 \wedge \cdots \wedge a_r) \cdot (b_1 \wedge \cdots \wedge b_s) = \begin{cases} ((a_1 \wedge \cdots \wedge a_r) \cdot b_1) \cdot (b_2 \wedge \cdots \wedge b_s) & \text{si } r \geq s \\ (a_1 \wedge \cdots \wedge a_{r-1}) \cdot (a_r \cdot (b_1 \wedge \cdots \wedge b_s)) & \text{si } r < s \end{cases} \tag{2.20}$$

y

$$(a_1 \wedge \cdots \wedge a_r) \cdot b_1 = \sum_{i=1}^{r} (-1)^{r-i} a_1 \wedge \cdots \wedge a_{i-1} \wedge (a_i \cdot b_1) \wedge a_{i+1} \wedge \cdots \wedge a_r \tag{2.21}$$

$$a_r \cdot (b_1 \wedge \cdots \wedge b_s) = \sum_{i=1}^{s} (-1)^{i-1} b_1 \wedge \cdots \wedge b_{i-1} \wedge (a_r \cdot b_i) \wedge b_{i+1} \wedge \cdots \wedge b_s \tag{2.22}$$

[2]Un vector **a** es nulo si $\mathbf{a}^2 = 0$

Consideremos la ecuación (2.20) con (2.22). Sean $\mathbf{a}, \mathbf{b}, \mathbf{c} \in \mathcal{G}_n$ 1-vectores y $\mathbf{B} = \mathbf{b} \wedge \mathbf{c}$ un 2-vector. El producto interior de $\mathbf{a}$ con $\mathbf{B}$ está dado por

$$\mathbf{a} \cdot \mathbf{B} = \mathbf{a} \cdot (\mathbf{b} \wedge \mathbf{c}) = (\mathbf{a} \cdot \mathbf{b})\mathbf{c} - (\mathbf{a} \cdot \mathbf{c})\mathbf{b} \tag{2.23}$$

Como $(\mathbf{a} \cdot \mathbf{b})$ y $(\mathbf{a} \cdot \mathbf{c})$ son escalares, podemos ver que el producto interior de un vector con un bivector resulta en un vector. Generalizando podemos mostrar que para $s \geq 1$

$$\begin{aligned} \mathbf{x} \cdot \mathbf{B}_{\langle s \rangle} &= (\mathbf{x} \cdot \mathbf{b}_1)(\mathbf{b}_2 \wedge \mathbf{b}_3 \wedge \mathbf{b}_4 \wedge \ldots \wedge \mathbf{b}_s) \\ &\quad -(\mathbf{x} \cdot \mathbf{b}_2)(\mathbf{b}_1 \wedge \mathbf{b}_3 \wedge \mathbf{b}_4 \wedge \ldots \wedge \mathbf{b}_s) \\ &\quad +(\mathbf{x} \cdot \mathbf{b}_3)(\mathbf{b}_1 \wedge \mathbf{b}_2 \wedge \mathbf{b}_4 \wedge \ldots \wedge \mathbf{b}_s) \\ &\quad -\ldots \\ &= \sum_{i=1}^{s} (-1)^{i-1} (\mathbf{x} \cdot \mathbf{b}_i)[\mathbf{B}_{\langle s \rangle} \backslash \mathbf{b}_i] \end{aligned} \tag{2.24}$$

donde $[\mathbf{B}_{\langle s \rangle} \backslash \mathbf{b}_i]$ denota el blade $\mathbf{B}_{\rangle}$ sin el vector $\mathbf{b}_i$. Así, el producto interior de un vector con un s-blade da como resultado un $(s-1)$-blade. Otro ejemplo de esta importante regla es

$$\mathbf{A} \cdot \mathbf{B}_{\langle s \rangle} = (\mathbf{a}_1 \wedge \mathbf{a}_2) \cdot \mathbf{B}_{\langle s \rangle} = \mathbf{a}_1 \cdot (\mathbf{a}_2 \cdot \mathbf{B}_{\langle s \rangle}) \tag{2.25}$$

con $s \geq 2$. Donde $\mathbf{A}$ es un 2-blade, $(r = 2)$. Si aumentamos el valor de r de tal forma que $r < s < n$ llegamos a

$$\mathbf{a}_1 \cdot \Big(\mathbf{a}_2 \cdot \big(\ldots \cdot (\mathbf{a}_r \cdot \mathbf{B}_{\langle s \rangle}) \big) \Big) \tag{2.26}$$

sr pr vemos que la Ecuación (2.26) es una representación de uno de los casos de la Ecuación (2.20).

Por bilinealidad, el producto interior es extendido hacia cualquier par de multivectores, si

$$\lambda \cdot M = M \cdot \lambda = 0, \text{ para } \lambda \in \mathbb{R}, M \in \mathcal{G}_n \tag{2.27}$$

Para cualquier blade A y B con grados r y s diferente de cero, tenemos

$$A \;\; B = \langle AB \rangle_{|r-s|} \tag{2.28}$$

$$A \wedge B = \langle AB \rangle_{r+s} \tag{2.29}$$

Mediante el producto geométrico podemos obtener blades de diferentes grados, el blade de más alto grado es el *n-blade* y es llamado "pseudoescalar" ($\mathtt{I} = \mathtt{e}_1 \wedge \mathtt{e}_2 \wedge \cdots \wedge \mathtt{e}_n$). Un AG se llama "no degenerada" si la magnitud del pseudo-escalar es diferente de cero. Los pseudo-escalares son indexados de acuerdo al álgebra a la que pertenecen, por ejemplo $\mathtt{I}_E \in \mathcal{G}_3$, $\mathtt{I}_C \in \mathcal{G}_{4,1}$[3].

El *dual* de un multivector M en $\mathcal{G}_n$ es denotado por

$$M^* = M\mathtt{I}_n^{-1} \tag{2.30}$$

donde $\mathtt{I}_n^{-1}$ difiere de $\mathtt{I}_n$ por lo menos por un signo. El "dual" de un *r-blade* es un *(n-r)-blade*.

La "reversión" de un *s-blade* $A = a_1 \wedge \cdots \wedge a_s$ se denota como

$$\begin{aligned} \widetilde{A} &= (a_1 \wedge a_2 \wedge \cdots \wedge a_{s-1} \wedge a_s)\widetilde{\ } \\ &= a_s \wedge a_{s-1} \wedge \cdots \wedge a_2 \wedge a_1 \end{aligned} \tag{2.31}$$

al generalizar para un multivector, la "reversión" es obtenida como

$$\langle \widetilde{M} \rangle_i = (-1)^{\frac{i(i-1)}{2}} \langle M \rangle_i, \text{ para } M \in \mathcal{G}_n, 0 \leq i \leq n. \tag{2.32}$$

Dado un producto geométrico de varios multivectores $D = ABC$, donde $A, B, C, D \in \mathcal{G}_n$, su reversión está dada por

$$\widetilde{D} = \widetilde{ABC} = \widetilde{C}\widetilde{B}\widetilde{A} \tag{2.33}$$

La "involución principal" de $\mathcal{G}_n$, también llamada "involución de grado" es denotada por "$\widehat{\ }$", y se define como

$$\langle \widehat{M} \rangle_i = (-1)^i \langle M \rangle_i, \text{ para } M \in \mathcal{G}_n, 0 \leq i \leq n. \tag{2.34}$$

El "producto conmutador" $A \times B$ es definido para cualquier multivector A y B por

$$A \times B \equiv \frac{1}{2}(AB - BA) = -B \times A. \tag{2.35}$$

Finalmente, podemos expresar la función exponencial de un multivector M como la expansión de la serie

$$exp(M) = \sum_{k=0}^{\infty} \frac{M^k}{k!} \tag{2.36}$$

[3]Para abreviar la notación $\mathcal{G}_{3,0,0} \simeq \mathcal{G}_3$ y $\mathcal{G}_{4,1,0} \simeq \mathcal{G}_{4,1}$

2.3. Álgebra geométrica del Espacio 3D Euclideano.

En muchos problemas de diferentes áreas es necesario trabajar con entidades geométricas. Por esta razón se requiere tener un álgebra que permita manejar diversas entidades (puntos, líneas, planos, etc.).

2.4. Rotores, trasladores y motores

En esta sección hablaremos de los movimientos rígidos de las entidades en el álgebra geométrica.

Los bivectores del AG pueden ser usados para representar rotaciones de entidades en el espacio de 3D. Un *rotor*[4] $\mathbf{R}$ es un elemento par del álgebra $\mathcal{G}^+_{3,0,1}$ (álgebra de motores) que satisface $\mathbf{R}\widetilde{\mathbf{R}} = 1$. En $\mathcal{G}_{3,0}$ una rotación puede ser expresada por un rotor de la forma

$$\mathbf{R} = exp\left(-\frac{\theta}{2}\underline{l}\right) \tag{2.37}$$

donde los componentes de este rotor $\mathbf{R} \in \mathcal{G}^+_{3,0,1}$ son similares al rotor de $\mathcal{G}_3$, un bivector unitario $\underline{l} \in \langle\mathcal{G}_3\rangle_2 \subseteq \mathcal{G}^+_{3,0,1}$, el cual representa una línea de magnitud unitaria que pasa por el origen (sin momento) que es el eje de rotación, y un ángulo θ, el cual representa el ángulo de rotación. Usando la representación de Euler del rotor (2.37) nos queda como

$$\begin{aligned} \mathbf{R} &= exp\left(-\frac{\theta}{2}\underline{l}\right) \\ &= cos\left(\frac{\theta}{2}\right) - sen\left(\frac{\theta}{2}\right)\underline{l} \end{aligned} \tag{2.38}$$

desarrollamos (2.38) para encontrar también a $\widetilde{\mathbf{R}}$

$$\mathbf{R} = r_c - r_s\underline{l} = r_c - r_{s_1}\mathbf{e}_{23} - r_{s_2}\mathbf{e}_{31} - r_{s_3}\mathbf{e}_{12} \tag{2.39}$$

$$\widetilde{\mathbf{R}} = r_c + r_s\underline{l} = r_c + r_{s_1}\mathbf{e}_{23} + r_{s_2}\mathbf{e}_{31} + r_{s_3}\mathbf{e}_{12} \tag{2.40}$$

La rotación de primitivas geométricas (punto, línea, plano en $\mathcal{G}_{3,0,0}$ o en $\mathcal{G}^+_{3,0,1}$) se realiza multiplicando $\mathbf{R}$ por la izquierda y por el reverso $\widetilde{\mathbf{R}}$ por la derecha.

[4]Rotor es una abreviación de *rotador*

Esto es, siendo L una línea, el resultado de una rotación será L' y viene dado por

$$L' = \mathbf{R}L\widetilde{\mathbf{R}} \tag{2.41}$$

Un rotor representa el grupo de SO(3) en el AG Euclidiana. El producto geométrico de dos rotores $\mathbf{R} = \mathbf{R}_1\mathbf{R}_2$ resulta en un nuevo rotor [1]. De esto se sigue que

$$L' = \mathbf{R}L\widetilde{\mathbf{R}} = (\mathbf{R}_2\mathbf{R}_1)L(\widetilde{\mathbf{R}_1}\widetilde{\mathbf{R}_2}) \tag{2.42}$$

Para trasladar una entidad geométrica respecto a un vector de traslación $\mathbf{t} \in \langle \mathcal{G}_3 \rangle_1$, usamos el *traslador*, $\mathbf{T} \in \mathcal{G}^+_{3,0,1}$

$$\mathbf{T} = exp\left(\frac{I\mathbf{t}}{2}\right) = (1 + \frac{I\mathbf{t}}{2}), I \in \mathcal{G}^+_{3,0,1} \tag{2.43}$$

utilizando (2.36) y la restricción $I^k = 0$ para $k \geq 2$. Un traslador es un rotor especial actuando en el infinito usando el pseudoescalar I. Similar a la rotación, una entidad puede ser trasladada multiplicando la entidad por $\mathbf{T}$ por la izquierda y por su reverso $\widetilde{\mathbf{T}}$ por la derecha.

$$X' = \mathbf{T}X\widetilde{\mathbf{T}} \tag{2.44}$$

Para expresar un movimiento rígido, la aplicación consecutiva de un rotor y un traslador pueden ser escritos como el producto de ellos. Tal operador es expresado como $\mathbf{M}$,

$$\mathbf{M} = \mathbf{TR} \tag{2.45}$$

Cinemática del punto, línea y plano en $\mathbf{G}^+_{3,0,1}$

Es un multivector especial de grado par llamado *motor*, el cual es la abreviación de *momento y vector*. El movimiento rígido por ejemplo de un punto X, línea L o plano Π puede ser escrito como

$$X = 1 \pm IX \tag{2.46}$$

$$X' = \mathbf{M}X\widetilde{\mathbf{M}} = \mathbf{TR}X\widetilde{\mathbf{R}}\widetilde{\mathbf{T}} \tag{2.47}$$

$$L' = \mathbf{M}L\widetilde{\mathbf{M}} = \mathbf{TR}L\widetilde{\mathbf{R}}\widetilde{\mathbf{T}} \tag{2.48}$$

$$\Pi' = \mathbf{M}\Pi\widetilde{\mathbf{M}} = \mathbf{TR}\Pi\widetilde{\mathbf{R}}\widetilde{\mathbf{T}} \tag{2.49}$$

Transformaciones de giro y tornillo

Siguiendo la definición del motor en el AG y basados en el rotor llamado de "giro", todo movimiento rígido puede ser expresado como un giro o movimiento de tornillo, el cual es una rotación sobre una línea en el espacio (en general, que no pase por el origen)[5] combinado con una traslación a lo largo de esta línea. En el AG es posible usar los rotores y trasladores para expresar movimientos de tornillo en el espacio.

Para modelar la rotación de un punto Q alrededor de una línea $\underline{L}$ arbitraria en el espacio, la idea general es trasladar el punto Q con el vector de distancia entre la línea $\underline{L}$ y el origen, para realizar la rotación en el origen y trasladar de regreso el punto transformado. Entonces el motor $\mathbf{M} \in \mathcal{G}^{+}_{3,0,1}$ describe una rotación general que tiene la forma

$$\mathbf{M} = \mathbf{T}\mathbf{R}\widetilde{\mathbf{T}} = \mathbf{R}_{\underline{L}} \tag{2.50}$$

donde se puede ver la traslación, la rotación y la traslación de regreso respectivamente. (2.50) se puede expresar como un rotor sobre la línea $\underline{L}$. Al aplicar $\mathbf{M} = \mathbf{R}\mathbf{L}$ a un punto obtenemos

$$Q' = \mathbf{R}_{\underline{L}} Q \widetilde{\mathbf{R}_{\underline{L}}} = \mathbf{T}\mathbf{R}\widetilde{\mathbf{T}} Q \mathbf{T}\mathbf{R}\widetilde{\mathbf{T}} \tag{2.51}$$

donde $\mathbf{T}\mathbf{R}\widetilde{\mathbf{T}}\widetilde{\mathbf{R}} = \widetilde{\mathbf{R}_{\underline{L}}}$. Usando la forma exponencial del traslador y rotor obtenemos[6]

$$\begin{aligned}
\mathbf{R}_{\underline{L}} &= \mathbf{T}\mathbf{R}\widetilde{\mathbf{T}} \\
&= exp\left(\frac{I\mathbf{t}}{2}\right) exp\left(-\frac{\theta}{2}\underline{l}\right) exp\left(-\frac{I\mathbf{t}}{2}\right) \\
&= \left(1 + \frac{I\mathbf{t}}{2}\right) exp\left(-\frac{\theta}{2}\underline{l}\right)\left(1 - \frac{I\mathbf{t}}{2}\right) \\
&= exp\left(\left(1 + \frac{I\mathbf{t}}{2}\right)\left(-\frac{\theta}{2}\underline{l}\right)\left(1 - \frac{I\mathbf{t}}{2}\right)\right) \\
&= exp\left(-\frac{\theta}{2}(\underline{l} + I(\mathbf{t} \cdot \underline{l}))\right)
\end{aligned} \tag{2.52}$$

Así, dada una línea en el AG que tenga magnitud unitaria, y usando su forma estándar $\underline{L}$, se puede hacer que cualquier entidad gire alrededor de ella

[5] Tal operador también es llamado "rotación general"

[6] En el paso 4 usamos la propiedad $\mathbf{g}exp(\xi)\widetilde{\mathbf{g}} = exp(\mathbf{g}\xi\widetilde{\mathbf{g}})$ para $\mathbf{g}\widetilde{\mathbf{g}} = 1$.

usando esta línea como eje de rotación. Esta expresión se puede representar como

$$\mathbf{R}_{\underline{L}} = exp\left(-\frac{\theta}{2}\underline{L}\right) \tag{2.53}$$

Los movimientos en tornillo pueden ser representados por movimientos rígidos. Un "movimiento de tornillo" es aquel que para cada movimiento rígido de cuerpos puede ser realizado por una rotación en un eje combinado con la traslación paralela en ese eje. Para modelar el movimiento de tornillo, la entidad tiene que trasladarse durante una rotación general respecto al eje de rotación. El grupo de un movimiento 3D rígido es $SE(3)$.

El motor resultante puede ser calculado utilizando (2.45) con (2.52) de la siguiente manera

$$\begin{aligned}
\mathbf{M} = \mathbf{T}_{d\mathbf{r}}\mathbf{R}_{\underline{L}} &= \mathbf{T}_{d\mathbf{r}}\mathbf{T}\mathbf{R}\widetilde{\mathbf{T}} \\
&= exp\left(\frac{\mathbf{e}d\mathbf{r}}{2}\right)exp\left(-\frac{\theta}{2}(\underline{l} + I(\mathbf{t}\cdot\underline{l}))\right) \\
&= exp\left(\frac{\mathbf{e}d\mathbf{r}}{2} - \frac{\theta}{2}(\underline{l} + I(\mathbf{t}\cdot\underline{l}))\right) \\
&= exp\left(-\frac{\theta}{2}\left(\underline{l} + I\underbrace{(\mathbf{t}\cdot\underline{l} - \frac{d}{\theta}\mathbf{r})}_{\mathbf{m}}\right)\right) \\
&= exp\left(-\frac{\theta}{2}(\underline{l} + I\mathbf{m})\right)
\end{aligned} \tag{2.54}$$

El bivector en la parte exponencial, $-\frac{\theta}{2}(\underline{l} + I\mathbf{m})$ es la representación de un tornillo. Si $\mathbf{m}$ es cero, el motor $\mathbf{M}$ actúa como un rotor y si $\underline{l}$ es cero, el motor actúa como un traslador. Para $\mathbf{m} \perp \underline{l}$, el motor actúa como una rotación general y para $\mathbf{m} \not\perp \underline{l}$, el motor actúa como un movimiento de tornillo.

2.4.1. Rotaciones en $\mathcal{G}_{3,0,0}$

Los rotores son usados para rotar objetos geométricos. La orientación de una línea está dada por $\mathbf{n}$ y la de un plano por $\mathbf{u}$ como se muestra en la figura 2.4. Las rotaciones de una línea y orientaciones de un plano en $\mathcal{G}_{3,0,0}$ están

dadas por:

$$\boldsymbol{n}' = \mathbf{R}\boldsymbol{n}\widetilde{\mathbf{R}} \tag{2.55}$$

$$\boldsymbol{u}' = \mathbf{R}\boldsymbol{u}\widetilde{\mathbf{R}} \tag{2.56}$$

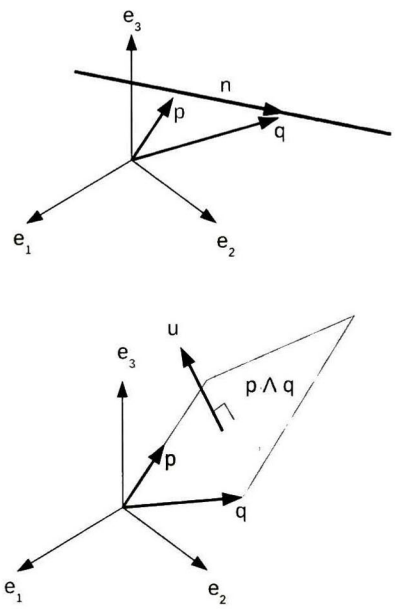


Figura 2.4: Línea y orientación de un plano en 3D

2.4.2. Motores en $\mathcal{G}^{+}_{3,0,1}$

Un "motor" puede ser encontrado en la subálgebra especial par de 4D de $\mathcal{G}_{3,0,1}$. Esta subálgebra par es denotada por $\mathcal{G}^{+}_{3,0,1}$ y es únicamente expandida vía una base de bivectores, como sigue:

$$\underbrace{1}_{\text{scalar}}, \underbrace{e_2e_3, e_3e_1, e_1e_2, e_4e_1, e_4e_2, e_4e_3,}_{\text{6 bivectors}} \underbrace{I}_{\text{unit pseudoscalar}} \tag{2.57}$$

Una transformación Euclidiana incluye tanto la rotación como la translación, y son representadas por un motor. Se observa que el dual de un escalar es

el pseudoescalar y que los duales de los primeros tres bivectores base son realmente los siguientes tres bivectores, esto es, ($e_2e_3* = Ie_2e_3 = e_4e_1$). Rotores y transladores son operadores específicos para modelar rotaciones y translaciones en AG. Los componentes del bivector de un motor son el rotor y el translator. Un simple rotor en su representación de *Euler* para una rotación con un ángulo θ satisface la relación 2.58,

$$\mathbf{R}\widetilde{\mathbf{R}} = 1 \tag{2.58}$$

donde

$$\mathbf{R} = \underbrace{u_0}_{scalar} + \underbrace{u_1\mathbf{e}_{23} + u_2\mathbf{e}_{31} + u_3\mathbf{e}_{12}}_{bivectors} \tag{2.59}$$

$$\widetilde{\mathbf{R}} = \underbrace{u_0}_{scalar} - \underbrace{u_1\mathbf{e}_{23} - u_2\mathbf{e}_{31} - u_3\mathbf{e}_{12}}_{bivectors} \tag{2.60}$$

El rotor puede ser expresado en forma exponencial. Tales operadores son definidos por la Eq. (2.61), donde **R** es el rotor, y **T** es el translador.

$$\mathbf{R} = e^{-\frac{1}{2}l\theta}; \qquad \mathbf{T} = e^{\frac{l\mathbf{t}}{2}}, \tag{2.61}$$

donde el "eje de rotación" $l = l_1\mathbf{e}_{23} + l_2\mathbf{e}_{31} + l_3\mathbf{e}_{12}$ es un bivector unitario, el cuál representa una línea a través del origen, θ es el ángulo de rotación y $\mathbf{t} = t_1\mathbf{e}_1 + t_2\mathbf{e}_2 + t_3\mathbf{e}_3$ es el vector de translación en $\mathbb{R}^3$ Las ecuaciones (2.61) pueden también ser expresadas como

$$\mathbf{R} = cos\left(\frac{\theta}{2}\right) - sin\left(\frac{\theta}{2}\right)\underline{l}; \qquad \mathbf{T} = (1 + \frac{I\mathbf{t}}{2}) \tag{2.62}$$

debido a las propiedades exponenciales. Tales operadores son aplicados a cualquier entidad de cualquier dimensión multiplicando la entidad por el operador desde la izquierda, y por la "reversa" del operador desde la derecha, como se indica:

$$x' = \sigma x\widetilde{\sigma} \tag{2.63}$$

donde x es una entidad geométrica y σ es un rotor, translator, o motor. Usando la Eq. (2.63), es sencillo transformar cualesquier entidad definida en AG, tales como puntos, líneas y planos. La implementación de un rotor y translador puede ser escrita como el producto de ellos. Tal operador es denominado motor y es definido como

$$\mathbf{M} = \mathbf{TR} \tag{2.64}$$

Un motor puede también ser expresado como

$$\begin{aligned} M &= TR = R + IR && (2.65)\\ &= (a_0 + a_1e_2e_3 + a_2e_3e_2 + a_3e_2e_1) \\ &\quad +I(b_0 + b_1e_2e_3 + b_2e_3e_2 + b_3e_2e_1) && (2.66)\\ &= (a_0 + \mathbf{a}) + I(b_0 + \mathbf{b}) && (2.67) \end{aligned}$$

El translador, rotor, y motor (todos ellos "versores") son elementos del AG $\mathcal{G}^+_{3,0,1}$, y definen un álgebra llamada "álgebra de motores" que simplifica el cálculo sucesivo de rotaciones y translaciones, aplicando el producto geométrico en consecutivos rotores, transladores o motores.

2.5. Movimiento con velocidad constante

La ecuación de movimiento con ruido se puede definir usando líneas en el álgebra de motores $\mathcal{G}^+_{1,3,0}$. Las características geométricas que se consideran son líneas observadas en 3D ($\mathbf{L}^1, \mathbf{L}^2, \ldots, \mathbf{L}^n, n \geq 2$) las cuales pertenecen a un objeto moviéndose en el espacio 3D. Los parámetros de movimiento rígido entre cualquier apareamiento de instantes de tiempo consecutivos $(t_0, t_1, t_2, \ldots, t_N)$ son descritos compactamente por el motor $\mathbf{M}_i$. El movimiento de cualquier línea de un objeto es modelado por

$$\mathbf{L}_i = \mathbf{M}_i\mathbf{L}_{i-1}\widetilde{\mathbf{M}}_i \tag{2.68}$$

Si el cambio de los parámetros de la línea en movimiento entre los instantes de tiempo $t_i - 1$ y t_i es descrito en términos de la velocidad $\mathbf{V}_{i/i-1}$,

$$\mathbf{L}_i = \mathbf{V}_{i/i-1}\mathbf{L}_{i-1}\widetilde{\mathbf{V}}_{i/i-1} \tag{2.69}$$

entonces se puede expresar la ecuación de movimiento recursivo de la línea en general como sigue:

$$\mathbf{L}_i = \mathbf{M}_i\mathbf{L}_{i-1}\widetilde{\mathbf{M}}_i = (\mathbf{V}_{i/i-1}\mathbf{M}_{i-1})\mathbf{L}_{i-1}(\widetilde{\mathbf{M}}_{i-1}\widetilde{\mathbf{V}}_{i/i-1}) \tag{2.70}$$

Por lo tanto, se obtiene el modelo de movimiento dinámico ideal en términos de los motores:

$$\mathbf{M}_i = \mathbf{V}_{i/i-1}\mathbf{M}_{i-1} \tag{2.71}$$

Considerando el movimiento tornillo con una rotación de velocidad angular constante, ω, alrededor de un eje de una línea conocida ($\mathbf{L}_s = \bar{\mathbf{r}} + \mathbf{I}\mathbf{t}_c \wedge \bar{\mathbf{r}}$) y con una velocidad de traslación constante, $\mathbf{v}_s$, a lo largo de la línea eje. Si el muestreo de los datos es hecho en intervalos de tiempo equidistantes, entonces los instantes de tiempo pueden ser representados por enteros, de manera que la ecuación del motor se lee como

$$\mathbf{V}_{i/i-1} = \mathbf{V} = (1 + \mathbf{I}\mathbf{v}_s/2)(cos(\omega/2) + sin(\omega/2)\mathbf{L}_s) \tag{2.72}$$

Dado que en un trabajo real, la relación entre $\mathbf{M}_{i-1}$ y $\mathbf{M}_i$ es conocida únicamente de forma aproximada, el modelo dinámico real del movimiento en 3D con ruido está dado por

$$\mathbf{M}_i = \mathbf{V}_{i/i-1,\mathbf{M}l}\mathbf{M}_{i-1} + \mathbf{W}_i \tag{2.73}$$

donde $\mathbf{W}_i$ es el error aleatorio.

Capítulo 3

NEUROCOMPUTACIÓN

Las redes neuronales artificiales (RNAs) son algoritmos bioinspirados en el funcionamiento de las neuronas biológicas, en ellas la eficiencia sináptica puede ser modelada como una propiedad de las conexiones de la red. La investigación se remonta a los diagramas de neuronas de Ramón y Cajal [27]. Las unidades McCulloch-Pitts usan únicamente señales binarias de manera que los nodos y las conexiones únicamente trabajan con unos o ceros. Cualquier función lógica puede ser calculada y cualquier autómata finito puede ser simulado con este tipo de unidades de cómputo. Las redes ponderadas y las no ponderadas son equivalentes, sin embargo con las ponderadas se resuelven problemas con menos unidades. El tipo de redes que se pueden construir con estas unidades McCulloch-Pitts no son muy relevantes [28]. Estas unidades de cómputo son muy similares a las compuertas lógicas convencionales. Las primeras RNAs tenían que ser completamente especificadas antes de usarse y carecían de parámetros libres que pudieran ser ajustados a diferentes problemas. El aprendizaje era implementado modificando el patrón de conexión de la red y los umbrales de las unidades, pero en realidad es necesariamente más complejo. Luego, se desarrollan las redes ponderadas. Rosenblatt propuso el perceptrón, un modelo computacional más general que el de McCulloch-Pitts. El perceptrón usa pesos reales y un patrón de interconexión especial. En este modelo la conectividad es determinada estocásticamente y las unidades son elementos con umbrales. El ajuste de los pesos se realiza con un algoritmo numérico. Posteriormente, Minsky y Papert modificaron el modelo. La única

diferencia entre el modelo de McCulloch-Pitts y el del perceptrón es la presencia de los pesos en las redes. La salida sigue siendo 1 ó 0. La interpretación geométrica del procesamiento realizado por los perceptrones es la misma que la de las unidades McCulloch-Pitts: separan el espacio de entrada en dos semiespacios. Se sabe que cuando algunos algoritmos son paralelizados, un componente secuencial irreductible limita la máxima velocidad alcanzable. La relación matemática entre velocidad y la porción secuencial irreductible de un algoritmo se denomina *Ley de Amdahl*. Existen problemas de reconocimiento de patrones en los que es necesario analizar secuencialmente la salida de los predicados asociados con cada unidad receptiva y que no pueden ser resueltos por un único perceptrón actuando como la última unidad de decisión. El "algoritmo de aprendizaje del perceptrón" básicamente hace una corrección del vector de pesos cuando uno de los vectores seleccionados de dos conjuntos dados no ha sido clasificado correctamente. Si estos conjuntos son linealmente separables, el vector **w** es actualizado únicamente un número finito de veces.

Resulta útil implementar algunas funciones no linealmente separables usando más de un perceptrón. Si el número y distribución de los conjuntos es desconocido, surge el problema de decidir la cantidad de unidades de cómputo y de vectores de pesos representativos a usarse ("problema de agrupación"). Este problema surge cuando se desea clasificar conjuntos de datos multidimensionales cuya estructura se deconoce. Aparece entonces el "algoritmo de aprendizaje competitivo", que contempla k unidades de cómputo de manera que los vectores de peso de las k unidades son "atraídos" en la dirección de los grupos en el espacio de entrada. La diferencia entre el algoritmo del perceptrón y éste último es que el conjunto de entrada no puede ser clasificado previamente en un conjunto negativo o positivo o en cualquiera de varias clases diferentes. Uno de los algoritmos de aprendizaje más populares para las RNA es el aprendizaje de Hebb. Esta regla es considerada como el principio básico de aprendizaje. Es una hipótesis relacionada con las redes neuronales biológicas, esto es, "cuando un axón de la célula A está lo suficientemente cerca como para excitar a la célula B y constantemente participa en dispararla, algún proceso de crecimiento o cambio metabólico toma lugar en una o ambas células tal que la eficiencia de A, como una de las células que hace disparar a B, es incrementada" Esta regla es expresada como el cambio temporal del

peso de conexión w_i como

$$\tau \frac{dw_i}{dt} = -w_i + yx_i \tag{3.1}$$

3.1. Redes de alimentación hacia adelante y recurrentes

Cuando las unidades de cómputo aisladas son combinadas para incrementar el poder de cómputo de la red, surgen entonces las redes que se definen de una manera más precisa en términos de su arquitectura.

3.1.1. Redes de alimentación hacia adelante

Sus elementos de procesamiento son las unidades de cómputo y sus interconexiones. Cada unidad de cómputo colecciona la información desde líneas de n entradas con una función de integración $\psi : \mathbb{R}^n \rightarrow \mathbb{R}$. La excitación total calculada de este modo se evalúa usando una función de activación $\varphi : \mathbb{R} \rightarrow \mathbb{R}$. Además se generaliza ϕ para producir todos los valores entre 0 y 1 (usando la función sigmoide), el continuo de valores de la salida puede ser comparado a una división del espacio de entrada en un continuo de clases. Estas redes pueden representar algunas funciones complejas con pocas unidades de cómputo. Estas redes no permiten ciclos de retroalimentación. La entrada es procesada y retransmitida de una capa a otra, hasta que el resultado final sea calculado. En estas redes, las unidades de una capa están conectadas a todas las otras unidades en la siguiente capa. Si hay m unidades en la primera capa y n unidades en la segunda, el número total de pesos es $m \times n$, entonces surge el concepto de "poda de la red". En una red neuronal de alimentación hacia adelante cada unidad de cómputo es capaz de evaluar una función primitiva única de su entrada. La red representa una cadena de composiciones de funciones que transforma una entrada a un vector de salida (llamada patrón). La red es una implementación particular de una función compuesta a partir del espacio de entrada al de salida, y podríamos denominarla "función de red"

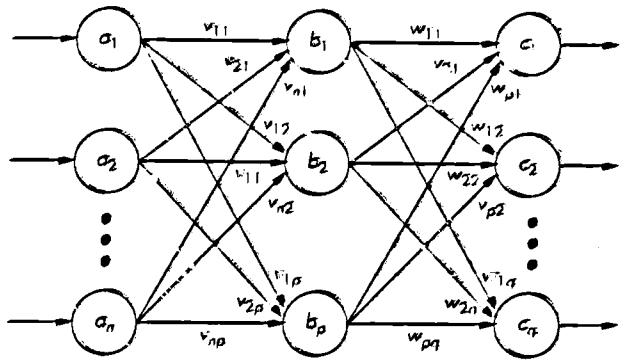


Figura 3.1: Red neuronal con capas

El problema del aprendizaje consiste en encontrar la combinación óptima de pesos tal que la función de red ρ aproxima una función dada f lo mejor posible. Sin embargo, no se da la función f explícitamente sino que de manera implícita a través de muestras. Las funciones primitivas en cada nodo de la red serán continuas y diferenciables. Los pesos de las aristas son valores determinados aleatoriamente. Cuando el patrón de entrada $\mathbf{x}_i$ del conjunto de entrenamiento es presentado a esta red, produce una salida $\mathbf{o}_i$ diferente en general del objetivo $\mathbf{t}_i$. Lo que se desea es lograr que $\mathbf{o}_i$ y $\mathbf{t}_i$ sean idénticos para $i = 1, \ldots, p$ usando un algoritmo de aprendizaje. Es decir, se desea minimizar la función de error de la red:

$$E = \frac{1}{2} \sum_{i=1}^{p} \| \mathbf{o}_i - \mathbf{t}_i \|^2 \tag{3.2}$$

Las redes multicapa son capaces de calcular un rango más amplio de funciones Booleanas que las redes con una única capa de unidades de cómputo. El esfuerzo computacional requerido para encontrar la combinación correcta de pesos en estas redes incrementa cuando se consideran más parámetros y topologías más complicadas. El "algoritmo de retro-propagación" puede tratar con estos problemas de entrenamiento largos. Éste busca el mínimo de la función de error en el espacio de pesos usando el gradiente descedente. Usa la función de activación sigmoide ya que si usara la "función escalón", la función compuesta producida por los perceptrones interconectados también sería discontinua y por lo tanto la función de error también. El algoritmo de retro-propagación es lento y se ha mejorado con técnicas de optimización no lineal. Surge entonces

la versión de "retro-propagación con momento" que implica usar para cada nueva combinación de pesos, un promedio ponderado de las direcciones del gradiente actual y la correción previa. Esto contrasta con la estrategia de únicamente seguir la dirección negativa del gradiente. Esto proveé cierta inercia al proceso de búsqueda y puede ayudar a abandonar excesivas oscilaciones en valles estrechos de la función de error. La retro-propagación se usa cuando no se tiene una expresión analítica de la función de error a ser optimizada. La función de error $E(\mathbf{w})$ es una función continuamente diferenciable de algún vector de parámetros. Esta función transforma los elementos de $\mathbf{w}$ en números reales, complejos o hipercomplejos, depende de la definición de la red. Para resolver este problema de optimización existe una clase de algoritmos basados en la idea de descendencia iterativa. Además del gradiente descedente, se usa el "método de Newton" cuya idea básica es minimizar la aproximación cuadrática de la función de costo $E(\mathbf{w})$ alrededor del actual vector de parámetros $w(k)$. También se usa el método de Gauss-Newton que únicamente requiere el Jacobiano del vector de error a diferencia del método de Newton que requiere el Hessiano de la función de costo. El algoritmo de mínimos cuadrados (LMS) se basa en el uso de los valores instantáneos de la función de costo. También, se usa el método de Levenberg-Marquardt que se basa en el método de multiplicadores de Lagrange. Este algoritmo interpola entre el algoritmo Gauss-Newton y el método del gradiente descendente, es más robusto que el primero lo que significa que en muchos casos encuentra una solución incluso si inicia muy lejos del mínimo global.

3.2. Redes recurrentes

Las redes de alimentación hacia adelante pueden implementar funciones lógicas arbitrarias. En este caso la dimensión de los datos de entrada y salida es predeterminada. En otros casos se desea desarrollar cálculos con una entrada de longitud variable. Este tipo de problemas puede ser resuelto por redes recurrentes cuyos cálculos parciales son reciclados a través de la misma red. Los ciclos en la topología de la red permiten el almacenamiento y reuso de señales en un cierto tiempo después de que son producidas. El retardo necesario en las

redes recurrentes para producir un resultado en tiempo $t+1$ cuando la entrada se presentó en el tiempo t, puede ser representado por elementos de cómputo adicionales.

En la red recurrente, existen conexiones entre unidades que forman un ciclo dirigido. Esto crea un estado interno de la red que permite exhibir un comportamiento dinámico temporal. A diferencia de las redes de alimentación hacia adelante, las redes recurrentes pueden usar su memoria interna para procesar secuencias de entradas. Esto las hace aplicables a tareas como el reconocimiento de escritura a mano conectada sin segmentar, donde han alcanzado los mejores resultados conocidos.

En una red recurrente simple (*SRN*), el vector de entrada es similarmente propagado a través de una capa de pesos, pero también combinado con la activación del estado previo mediante una capa de pesos recurrentes adicionales,

$$y_j(t) = f(net_j(t)) \tag{3.3}$$

$$net_j(t) = \sum_i^n x_i(t)v_{ji} + \sum_h^m y_h(t-1)u_{jh}) + \theta_j \tag{3.4}$$

donde m es el número de nodos de "estado", n es el número de entradas, θ_j es el *bias* y f es una función de salida (de cualquier tipo diferenciable). Cada capa tiene su propia variable índice: k para los nodos de salida, j (y h) para los nodos ocultos, e i para los nodos de entrada. El vector de entrada, $\mathbf{x}$ es propagado a través de una capa de pesos $\mathbf{V}$.

La salida de la red está determinada por el estado y un conjunto de pesos de salida, $\mathbf{W}$,

$$y_k(t) = g(net_k(t)) \tag{3.5}$$

$$net_k(t) = \sum_j^m y_j(t)w_{kj} + \theta_k \tag{3.6}$$

donde g es una función de salida (posiblemente la misma que f).

3.2.1. Representación del tiempo en redes neuronales

La representación del tiempo dada en [29, 30], permite visualizar dos tipos de soluciones (Figura 3.2). El tiempo en redes neuronales puede ser representado por un mecanismo interno o externo. Estos dos términos corresponden respectivamente a una representación espacial y dinámica del tiempo [31]. En la representación espacial, el tiempo es introducido en el modelo con la ayuda de un mecanismo externo. El objetivo es encontrar arquitecturas particulares que permitan ejecutar los parámetros dinámicos. Las redes conocidas usando realmente este concepto son: *TDNN (Time Delay Neural Network)* usada con el famoso software de reconocimiento de voz NetTalk, y la *TDRBF (Time Delay Radial Basis Function)* usada para el reconocimiento de fonemas [32]. El mayor inconveniente de estos algoritmos es la existencia de una interface externa con el ambiente con el ambiente para retrasar y guardar los datos. La segunda desventaja es el uso de una ventana temporal que impone un límite de la longitud de la secuencia. La mayor ventaja de las redes *TDRBF* en comparación a la *TDNN* es la flexibilidad del entrenamiento, y el número reducido de parámetros para ajustar el tiempo de entrenamiento [32].

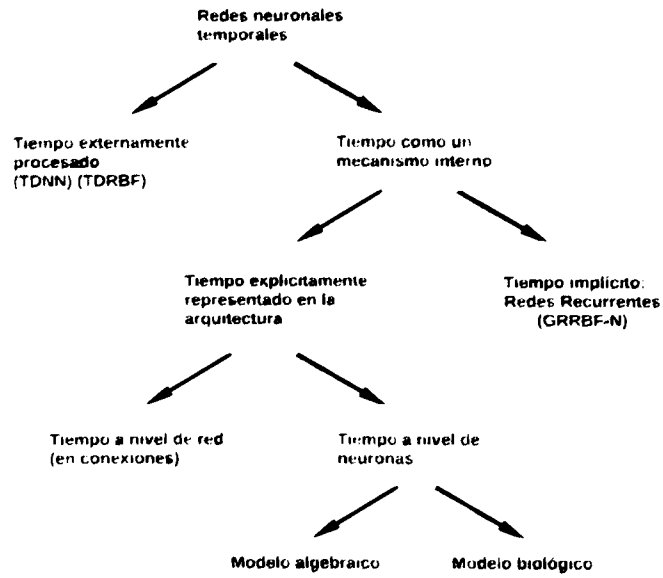


Figura 3.2: Redes temporales

En cuanto a la representación dinámica del tiempo, cuando el tiempo es un mecanimo interno, el tiempo puede ser implícito (redes recurrentes) o explícito (la variable tiempo aparece en las conexiones de la red o neuronas). Para el tiempo explícito a nivel neuronal (el enfoque más "biológico"), existen métodos usando modelos algebraicos o biológicos. Ambos métodos usan algoritmos muy complicados y requieren software sofisticado para su ejecución. Por esta razón, nuestros estudios se enfocan en la clase de redes neuronales de tiempo implícito.

Este problema consiste en producir redes que exhiban un comportamiento temporal que se desea.

3.2.2. El modelo Elman

En esta investigación se contemplan variantes de los modelos Elman y Jordan. El modelo de Elman clásico se basa en una arquitectura básica de red recurrente. Consiste de tres capas, con la adición de un conjunto de "unidades de contexto" en la capa de entrada. Existen conexiones desde la capa oculta a estas unidades de contexto ponderadas con un peso cuyo valor es uno. En cada paso temporal, la entrada es propagada de la manera clásica de retroalimnetación hacia adelante, y se aplica una regla de aprendizaje. Las conexiones de respaldo fijo resultan en que las unidades de contexto siempre mantienen una copia de los valores previos de las unidades ocultas (dado que se propagan sobre las conexiones antes de que la regla de aprendizaje es aplicada en ese instante). Por lo tanto, la red puede mantener una especie de estado, permitiendole llevar a cabo tareas como predicción de secuencias que están más allá del poder de un perceptrón multicapa estándar.

3.2.3. El modelo Jordan

Este modelo es similar al anterior, sin embargo las unidades de contexto son alimentadas desde la capa de salida en vez de la capa oculta. Éstas redes son conocidas como "redes recurrentes simples" (*SRN*). Una arquitectura de estricta alimentación hacia adelante no mantiene una memoria de corto plazo.

Cualquier efecto de una memoria es debido a la manera en que las entradas pasadas son presentadas nuevamente a la red. El caso del modelo de Elman tiene retro-alimentación de activación que implica una memoria de corto plazo. Una capa de estado es actualizada no únicamente con la entrada externa de la red sino también con la activación de la previa propagación hacia adelante. La retroalimentación es modificada por un conjunto de pesos para permitir una adaptación automática a través del aprendizaje.

3.2.4. Gradiente descendente

Para minimizar el error total, el gradiente descendente puede ser usado para cambiar cada peso en proporción a su derivada con respecto al error, siempre y cuando las funciones de activación no lineales sean diferenciables. Existen muchas variantes de este algoritmo [Paul Werbos, Ronald J. Williams, Tony Robinson etc]. El algoritmo de retro-propagación también se extendió al caso de estas redes: "retro-propagación a través del tiempo" y es el método estándar. Existe una variante en línea computacionalmente más cara y se denominada "Aprendizaje Recurrente en Tiempo Real (*Real-Time Recurrent Learning* o RTRL)" Un problema importante con el gradiente descedente para las arquitecturas estándares de Redes Neuronales Recurrentes es que los gradientes del error se desvanecen exponencialmente rápido con el tamaño de tiempo de retrazo entre eventos importantes.

3.2.5. Métodos de optimización globales

El entrenamiento de los pesos en una red neuronal es un problema de optimización global no lineal. Una función objetivo puede ser formada para evaluar la aptitud o error de un vector de pesos específico de la siguiente manera: primero, los pesos en la red se establecen de acuerdo con el vector de pesos. Luego, la red se evalúa con respecto a la secuencia de entrenamiento. Típicamente, se usa la suma de los cuadrados de diferencias entre las predicciones y los valores objetivo especificados en la secuencia de entrenamiento para representar el error del vector de peso actual. Existen diversas técnicas

de optimización global que pueden ser usadas para minimizar esta función objetivo. El método de optimización global más común para entrenar redes neuronales recurrentes es el de algoritmos genéticos, especialmente en redes no estructuradas. Pueden usarse otras técnicas de optimización globales para buscar un conjunto adecuado de pesos, tales como recocido simulado u optimización por nubes de partículas.

3.3. RNAs valuadas con números hiper-complejos

Con el objetivo de modelar el cerebro artificial, es necesario considerar nuevas estructuras de redes neuronales que permitan almacenar y procesar información cuya representación no puede darse con valores reales. Estas nuevas estructuras únicamente pueden ser creadas en el marco del álgebra geométrica. Este tipo de redes permite que para cada cierto problema se adopte una arquitectura específica. Cada módulo del cerebro artificial estaría representado por una arquitectura neuronal, posiblemente complicada, que pueda trabajar con cierto tipo de datos específicos (Figura 3.3).

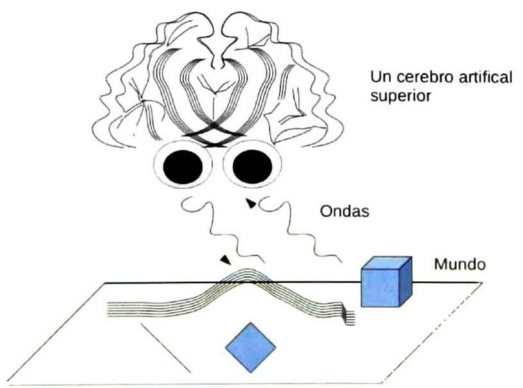


Figura 3.3: Cerebro artificial

3.4. Problema relacionado con funciones de activación en RNA valuadas con números hipercomplejos

Durante la construcción de estos nuevos modelos, un problema surge cuando extendemos la función sigmoide a una función de saturación con valores hipercomplejos para construir una RNVH, analicemos el caso de los complejos con el objetivo de tener mayor claridad. El problema consiste en que tal función no lineal no puede ser analítica. Una función compleja es llamada "analítica" cuando es diferenciable en cualquier punto. Una función compleja es diferenciable en un punto u en el plano complejo si podemos determinar el valor del límite de $\frac{(f(u+\Delta u)-f(u))}{\Delta u}$ cuando $\Delta u \to 0$ para la función $f(u)$ de la variable compleja u, el valor del límite no depende de la dirección en la cual la variable se aproxima al punto u.

Cuando una función considerada es diferenciable en un punto, podemos decir que la función es regular en el punto y si en ese punto la función no es diferenciable se tiene un punto singular. Una función es analítica si es regular en cualquier punto en el dominio considerado. Si la función de activación es una función analítica, podemos analizar las dinámicas neuronales tales como aprendizaje, autoorganización, y procesamiento, para entender las características de las RNVHs de la misma manera en la que se investiga la dinámica de las redes neuronales convencionales.

Consideremos que la variable u (Ecuación 3.7) es compleja. Entonces la función $f(u)$ es diferenciable en casi cualquier punto, pero diverge al infinito. Por lo tanto difiere mucho de la característica de "saturación" Aunque la $tanh(u)$ compleja en el dominio complejo puede ser una extensión natural de la $tanh(u)$ valuada en reales en el sentido de que la variable es extendida en una compleja, el significado de la nolinealidad es completamente diferente de la saturación. Por lo tanto, no podemos construir un sistema útil con la $tanh(u)$ compleja. Es ahora evidente que este problema fue la causa más seria por la

que las RNVCs fueran consideradas difíciles de desarrollar en el pasado.

$$y = f(u) = f\left(\sum_{i=1}^{N} w_i x_i\right) \tag{3.7}$$

3.4.1. Redes neuronales valuadas con reales

La aproximación de "mapeos no lineales" usando redes neuronales es útil en varios aspectos de procesamiento de señales, tales como la clasificación de patrones, predicción, modelado de sistemas, e identificación. Primero surgieron las arquitecturas estándares de "alimentación hacia adelante" valuadas con reales. En este tipo de estructuras, Cybenko [33] usó la superposición de funciones ponderadas para la aproximación de una función continua $g(\mathbf{x})$

$$y(\mathbf{x}) = \sum_{j=1}^{N} w_j \sigma_j(\mathbf{w}_j^T \mathbf{x} + \theta_j) \tag{3.8}$$

donde $\sigma(\cdot)$ es una función continua discriminatoria como un sigmoide, $w_j \in \mathbb{R}$ y $\mathbf{x}, \theta_j, \mathbf{w}_j \in \mathbb{R}^n$

Una estructura con k salidas y_k, teniendo varias capas usando la función logística, se conoce como "perceptrón multicapa" (MLP) [34]. La salida de cualquier neurona de una capa oculta o de la capa de salida puede ser representada de manera similar

$$o_j = f_j(\sum_{j=1}^{N_i} w_{ji} x_{ji} + \theta_j) \qquad y_k = f_k(\sum_{j=1}^{N_j} w_{kj} o_{kj} + \theta_k) \tag{3.9}$$

donde $f_j(\cdot)$ es logística y $f_k(\cdot)$ es logística o lineal. Las funciones lineales en las salidas son usadas frecuentemente para clasificación de patrones. En algunas tareas de clasificación de patrones, una capa oculta es necesaria, mientras que en algunas tareas de control automático, se pueden requerir dos capas ocultas. En [35] se mostró que los MLPs estándares son capaces de aproximar precisamente cualquier función medible con cierto grado deseado. Por esto, los MLPs son "aproximadores universales" En el caso de que el entrenamiento falle, cualquier error se puede atribuir a un aprendizaje inadecuado, un número incorrecto de neuronas ocultas, o una relación determinista pobremente definida

entre los patrones de entrada y de salida. Posteriormente, se desarrolló [36] la red de "funciones de base radial" (RBF) que consiste de una superposición de funciones Gausianas ponderadas

$$y_j(\mathbf{x}) = \sum_{i=1}^{N} w_{ji} G_i(D_i(\mathbf{x} - \mathbf{t}_i)) \tag{3.10}$$

donde y_j es la salida j, $w_{ji} \in \mathbb{R}$, G_i es una función Gausiana, D_i una matriz diagonal de dilatación $N \times N$, y $\mathbf{x}, \mathbf{t}_i \in \mathbb{R}^n$. El vector $\mathbf{t}_i$ es un vector de traslación. Esta arquitectura está fundamentada en la teoría de regularización.

3.4.2. El MLP complejo y el MLP Cuaterniónico

Un MLP se encuentra definido en el dominio complejo cuando sus pesos, función de activación, y salidas son valuadas con complejos. La selección de la función de activación no es un asunto trivial. La extensión de la función sigmoide de $\mathbb{R}$ a $\mathbb{C}$

$$\mathbf{f}(\mathbf{z}) = \frac{1}{(1 + e^{-\mathbf{z}})} \tag{3.11}$$

donde $\mathbf{z} \in \mathbb{C}$, no se permite porque esta función es analítica y no acotada [37], esto también se cumple con las funciones $tanh(\mathbf{z})$ y $e^{-\mathbf{z}^2}$. Este tipo de funciones de activación exhiben problemas con la convergencia en el entrenamiento debido a sus singularidades. Las condiciones necesarias que una $\mathbf{f}(\mathbf{z}) = a(x, y) + ib(x, y)$ de activación compleja debe cumplir son: $\mathbf{f}(\mathbf{z})$ debe ser no lineal en x y y, las derivadas parciales a_x, a_y, b_x y b_y deben existir $(a_x b_y \not\equiv b_x a_y)$ y $\mathbf{f}(\mathbf{z})$ no debe ser entera. De acuerdo a esto, se propuso en [37] la formulación

$$\mathbf{f}(\mathbf{z}) = \frac{\mathbf{z}}{c + \frac{1}{r}|\mathbf{z}|} \tag{3.12}$$

donde $c, r \in \mathbb{R}^+$. Estas investigaciones extendieron la regla de aprendizaje tradicional de retro-propagación valuada con reales a la regla valuada con complejos del "perceptrón multicapa complejo" (CMLP). Arena et al. [38] introdujeron el perceptrón multicapa cuaterniónico (QMLP) el cual es una extensión del CMLP. Los pesos, las funciones de activación, y las salidas de esta red son representadas en términos de cuaterniones [39]. Arena et al. seleccionaron la

siguiente función acotada no analítica

$$\begin{aligned} \mathbf{f(q)} &= \mathbf{f}(q_0 + q_1 i + q_2 j + q_3 k) \\ &= (\frac{1}{1+e^{-q_0}}) + (\frac{1}{1+e^{-q_1}})i + (\frac{1}{1+e^{-q_2}})j + (\frac{1}{1+e^{-q_3}})k \end{aligned} \tag{3.13}$$

donde $\mathbf{f}(\cdot)$ es la función para cuaterniones. La extensión de la regla de entrenamiento para el CMLP fue demostrada en [38].

3.4.3. Redes neuronales con álgebra geométrica

Las redes neuronales reales, complejas y cuaternionicas pueden ser generalizadas dentro del marco de trabajo del álgebra geométrica (AG), en la que los pesos, las funciones de activación y las salidas son ahora representados usando "multivectores". Para las redes neuronales valuadas con reales discutidas en la sección 3.4.1, los vectores son multiplicados con los pesos, usando el producto escalar. Para las redes neuronales geométricas, el producto escalar es reemplazado por el producto geométrico. La importancia de aplicar el álgebra geométrica en la arquitectura de las redes neuronales radica en que las entidades definidas con esa álgebra pueden ser manipuladas de manera directa con este tipo de algoritmos debido a que internamente existen operadores de transformación que pueden procesar la información de entrada para tener resultados en el mismo marco de trabajo del álgebra geométrica. Además, con esta álgebra es posible considerar números complejos, cuaterniones, o de alguna dimensión mayor sin cambiar la arquitectura de las redes neuronales geométricas.

3.4.4. La función de activación

La función de activación de la ecuación 3.12, usada por el CMLP, fue extendida por Pearson and Bisset [40] para un tipo de Clifford MLP aplicando diferentes algebras de Clifford, incluyendo el álgebra de cuaterniones. También, en [1] se propuso una función de activación que afecta cada elemento de base multivectorial. Esta función fue introducida independientemente por los autores en [9] y es de hecho una generalización de la función de Arena et al.

[38]. La función para un multivector $n-$dimensional está dada por

$$\begin{aligned}\mathbf{f(m)} &= \mathbf{f}(m_0 + m_i\sigma_i + m_j\sigma_j + m_k\sigma_k + \cdots + m_{ij}\sigma_i \wedge \sigma_j + \cdots + \\ &\quad m_{ijk}\sigma_i \wedge \sigma_j \wedge \sigma_k + \cdots + m_n\sigma_1 \wedge \sigma_2 \wedge \cdots \wedge \sigma_n) \\ &= (m_0) + f(m_i)\sigma_i + f(m_j)\sigma_j + f(m_k)\sigma_k + \cdots + f(m_{ij})\sigma_i \wedge \sigma_j + \cdots + \\ &\quad f(m_{ijk})\sigma_i \wedge \sigma_j \wedge \sigma_k + \cdots + f(m_n)\sigma_1 \wedge \sigma_2 \wedge \cdots \wedge \sigma_n \end{aligned} \tag{3.14}$$

Los valores de $f(\cdot)$ pueden ser de tipo Gausiano o sigmoide.

3.4.5. La neurona geométrica

La "neurona" o unidad de cómputo modelada por McCulloch-Pitts usa el producto escalar del vector de entrada y su vector de pesos [34]. La extensión de este modelo a la "neurona geométrica" requiere sustituir el producto escalar con el producto geométrico, es decir

$$\mathbf{w}^T\mathbf{x} + \theta \qquad \Rightarrow \qquad \mathbf{wx} + \boldsymbol{\theta} = \mathbf{w} \cdot \mathbf{x} + \mathbf{w} \wedge \mathbf{x} + \boldsymbol{\theta} \tag{3.15}$$

La figura 3.4 muestra las neuronas McCulloc-Pitts y la geométrica. Esta figura también hace evidente cómo el patrón de entrada es formateado en un AG específica. La neurona geométrica genera un tipo de patrón más rico en información.

Se puede ilustrar esto con un ejemplo en $G_{3,0,0}$

$$\begin{aligned}\mathbf{o} &= \mathbf{f(wx + \theta)} \\ &\quad \mathbf{f}(a_0 + a_1\sigma_1 + a_2\sigma_2 + a_3\sigma_3 + a_4\sigma_1\sigma_2 + a_5\sigma_1\sigma_3 + a_6\sigma_2\sigma_3 + a_7\sigma_1\sigma_2\sigma_3) \\ &= f(a_0) + f(a_1)\sigma_1 + f(a_2)\sigma_2 + f(a_3)\sigma_3 + f(a_4)\sigma_1\sigma_2 + \cdots + \\ &\quad f(a_5)\sigma_1\sigma_3 + f(a_6)\sigma_2\sigma_3 + f(a_7)\sigma_1\sigma_2\sigma_3 \end{aligned} \tag{3.16}$$

donde **f** es la función de activación definida en la ecuación 3.14, y $a_i \in \mathbb{R}$. Si se usa la neurona de tipo McCulloch-Pitts en la red neuronal valuada con reales, la salida es simplemente el escalar dado por

$$o = f(\sum_{i=1}^{N} w_i x_i + \theta) \tag{3.17}$$

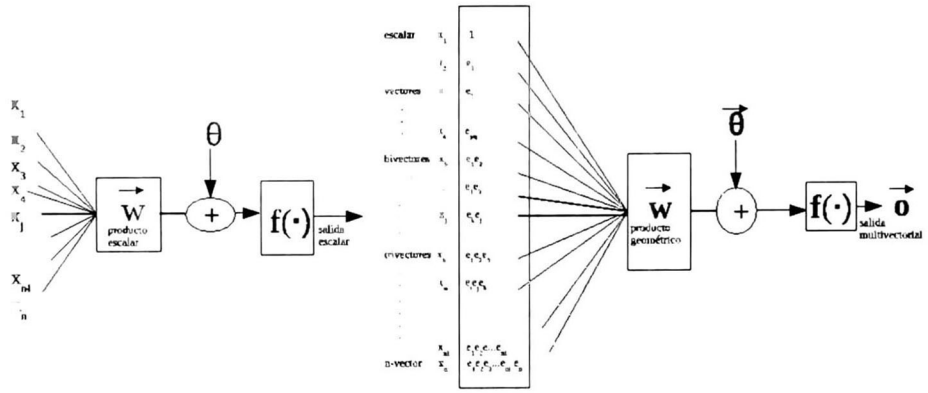


Figura 3.4: Las neuronas de tipo McCulloch-Pitts y geométrica

La neurona geométrica emite una señal con más información geométrica

$$\mathbf{o} = \mathbf{f}(\mathbf{wx} + \boldsymbol{\theta}) = \mathbf{f}(\mathbf{w} \cdot \mathbf{x} + \mathbf{w} \wedge \mathbf{x} + \boldsymbol{\theta}) \tag{3.18}$$

Tiene el producto escalar que calcula la neurona de tipo McCulloch-Pitts

$$\mathbf{f}(\mathbf{w} \cdot \mathbf{x} + \theta) = f(a_0) \equiv f(\sum_{i}^{N} w_i x_i + \theta) \tag{3.19}$$

y también el producto exterior dado por

$$\begin{aligned} \mathbf{f}(\mathbf{w} \wedge \mathbf{x} + \boldsymbol{\theta} - \theta) \;=\; & f(a_1)\sigma_1 + f(a_2)\sigma_2 + f(a_3)\sigma_3 + f(a_4)\sigma_1\sigma_2 + \cdots + \\ & f(a_5)\sigma_1\sigma_3 + f(a_6)\sigma_2\sigma_3 + f(a_7)\sigma_1\sigma_2\sigma_3 \end{aligned} \tag{3.20}$$

Se observa que el producto exterior calcula los productos cruz de tipo escalar entre los componentes individuales del vector, que no son nada más que los componentes del multivector de puntos y líneas (vectores), planos (bivectores) y volúmenes (trivectores). Esta característica puede ser usada para implementar el preprocesamiento geométrico en la red neuronal geométrica extendida. De cierto modo, este tipo de red neuronal se asemeja a las redes neuronales de alto orden [41]. Sin embargo, una red neuronal geométrica extendida no únicamente usa el producto escalar de alto orden, también usa todos los productos

cruz de tipo escalar necesarios para lograr una "correlación cruz geométrica" La figura 3.5 muestra una red geométrica con su primera capa extendida. Una neurona geométrica puede ser vista como un tipo de "operador de correlación geométrico", el cual, en contraste a la neurona de tipo McCulloch-Pitts, no ofrece únicamente puntos sino también multivectores de grado superior tales como planos, volúmenes, ..., e hipervolúmenes para la interpolación.

3.4.6. Redes neuronales geométricas de alimentación hacia adelante

En [1] se muestran algunas estructuras de redes neuronales estándares para la aproximación de funciones en el marco de trabajo del AG. En ellas, el producto interior de vectores ha sido extendido al producto geométrico y las funciones de activación son definidas como en la ecuación 3.14. La ecuación (3.8) del modelo de Cybenko en el marco del AG resulta como

$$\mathbf{y}(\mathbf{x}) = \sum_{j=1}^{N} \mathbf{w}_j \mathbf{f}(\mathbf{w}_j \cdot \mathbf{x} + \mathbf{w}_j \wedge \mathbf{x} + \boldsymbol{\theta}_j) \tag{3.21}$$

La extensión del MLP es sencilla. Las ecuaciones usando el producto geométrico para las salidas de las capas oculta y de salida están dadas por

$$\mathbf{o}_j = \mathbf{f}_j(\sum_{i=1}^{N_i} \mathbf{w}_{ji} \cdot \mathbf{x}_{ji} + \mathbf{w}_{ji} \wedge \mathbf{x}_{ji} + \theta_j) \tag{3.22}$$

$$\mathbf{y}_k = \mathbf{f}_k(\sum_{j=1}^{N_j} \mathbf{w}_{kj} \cdot \mathbf{o}_{kj} + \mathbf{w}_{kj} \wedge \mathbf{o}_{kj} + \theta_k) \tag{3.23}$$

En [1], se propone que en las redes de función de base radial (RBF-N), la operación de dilatación, dada por la matriz diagonal D_i, puede ser implementada por medio del producto geométrico con una dilatación $\mathbf{D}_i = e^{\alpha \frac{\vec{n}}{2}}$ [42], es decir

$$D_i(\mathbf{x} - \mathbf{t}_i) \qquad \Rightarrow \qquad \mathbf{D}_i(\mathbf{x} - \mathbf{t}_i)\tilde{\mathbf{D}}_i \tag{3.24}$$

$$\mathbf{y}_k(\mathbf{x}) = \sum_{j=1}^{N} \mathbf{w}_{kj} \mathbf{G}_j(\mathbf{D}_j(\mathbf{x}_{ji} - \mathbf{t}_j)\tilde{\mathbf{D}}_j) \tag{3.25}$$

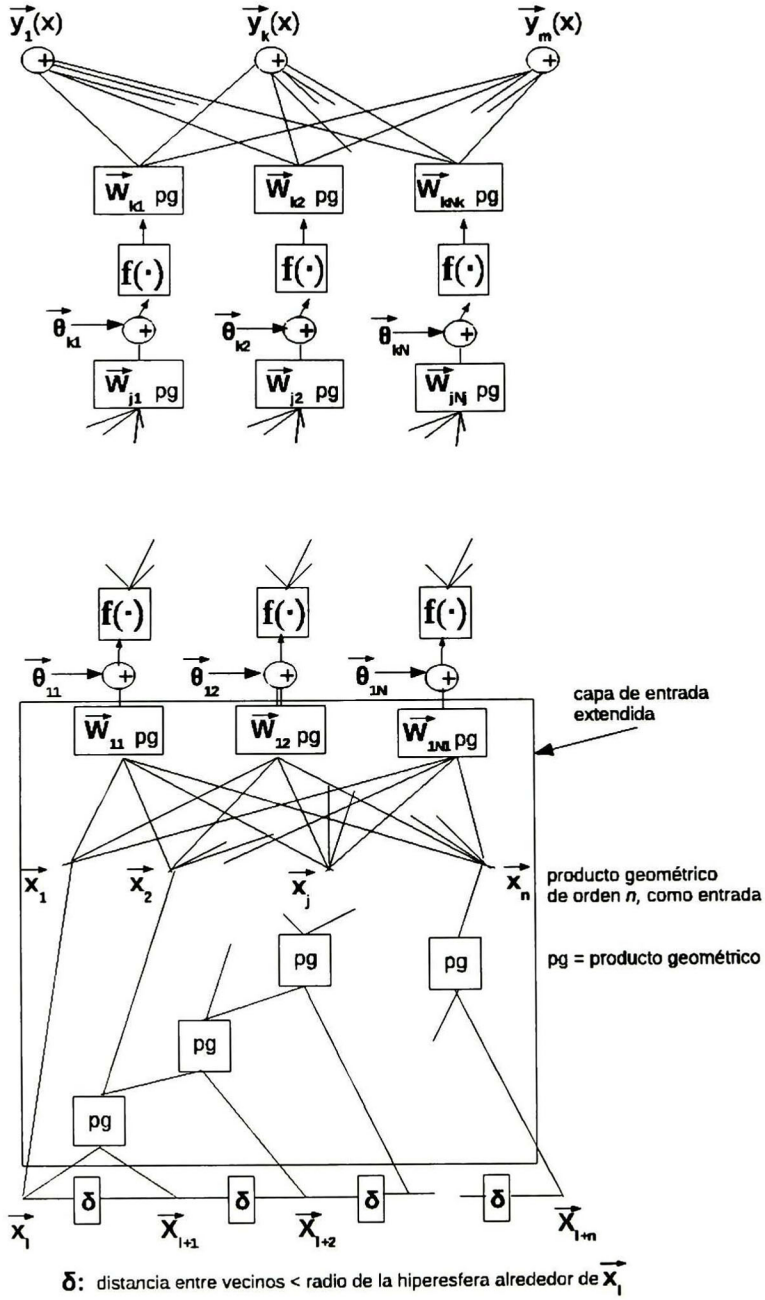


Figura 3.5: Red neuronal geométrica con la capa de entrada extendida [1].

Se puede apreciar que en el caso de la RBF-N geométrica se está usando también una función de activación de acuerdo a la ecuación 3.14. La ecuación 3.25 con $\mathbf{w}_{kj} \in \mathbb{R}$ representa la ecuación de una arquitectura de RBF-N para multivectores de dimensión 2^n, la cual es isomorfa a una RBF-N valuada con reales y con vectores de entrada de dimensión 2^n

3.4.7. Redes neuronales geométricas generalizadas

Una de las mayores ventajas del uso del AG en neurocomputación es que las redes trabajan para todo tipo de multivectores: reales, complejos, dobles (o hiperbólicos), y duales, asi como también para diferentes tipos de modelos de cómputo, como horósferas y conos nulos ([43, 26]). La base multivectorial seleccionada para una AG particular $G_{p,q,r}$ define la signatura de los subespacios involucrados. La signatura es calculada formando el cuadrado del pseudoescalar: si $I^2 = -1$, la red usará números complejos; si $I^2 = 1$, la red usará números dobles o hiperbólicos; y si $I^2 = 0$, la red usará números duales (una AG degenerada).

En el caso de $G_{0,2,0}$ se puede tener una red neuronal valuada con cuaterniones; para $G_{1,1,0}$, un MLP hiperbólico; para $G_{0,3,0}$, una red neuronal RBF hiperbólica (doble) valuada con cuaterniones: o para $G_{3,0,0}$ se puede tener una red que trabaje en la completa AG tridimensional Euclidiana; para $G^+_{4,1,0}$, se puede tener una red que trabaje en la horósfera; o finalmente, para $G^+_{3,3,0}$ se puede tener una red que use únicamente el cono nulo bivectorial. La conjugación involucrada en la regla de aprendizaje durante el entrenamiento, depende del tipo de valor, es decir, si se están usando redes neuronales geométricas valuada con complejos, con hiperbólicos o duales, y varía de acuerdo a la signatura del álgebra geométrica.

3.5. La regla de aprendizaje

En [1], se demuestra la generalización multidimensional de la regla de aprendizaje del gradiente descendente en el marco de trabajo del AG. Esta

regla puede ser usada para el entrenamiento del MLP geométrico (GMLP) y para la sintonización de los pesos de la RBF-N geométrica (GRBF-N). Las reglas previas de aprendizaje para el MLP valuado con reales, el MLP valuado con complejos [37] y el MLP cuaterniónico [38] son casos especiales de esta regla extendida.

3.5.1. Regla de aprendizaje de retro-propagación multidimensional

La "norma de un multivector" x para la regla de aprendizaje está dada por

$$|\mathbf{x}| = (\mathbf{x}|\mathbf{x})^{\frac{1}{2}} = (\sum_{\mathbf{A}} [\mathbf{w}]_{\mathbf{A}}^2)^{\frac{1}{2}} \tag{3.26}$$

La red neuronal geométrica con n entradas y m salidas aproxima la función de mapeo objetivo

$$y_t : (G_{p,q,r})^n \rightarrow (G_{p,q,r})^m \tag{3.27}$$

donde $(G_{p,q,r})^n$ es el módulo n-dimensional sobre el AG $G_{p,q,r}$ [40]. El error en la salida de la red es medido de acuerdo a la métrica

$$E = \frac{1}{2} \int_{\mathbf{x} \in \mathbf{X}} |y_w - y_t|^2 \tag{3.28}$$

donde $\mathbf{X}$ es algún subconjunto compacto del módulo Clifford $(G_{p,q,r})^n$ involucrando la topología de producto derivada de la ecuación 3.26 para la norma y donde y_w y y_t son las funciones de mapeo aprendida y objetivo, respectivamente. El "algoritmo de retro-propagación" [34] es un procedimiento para actualizar los pesos y el sesgo (bias). Este algoritmo es una función de la derivada negativa de la función de error (Ecuación 3.28) con respecto a los mismos pesos y sesgo (bias). El cálculo de este procedimiento es sencillo, y a continuación se dan los principales resultados [1]. La ecuación de actualización para los pesos multivectoriales de cualquier capa j oculta es

$$\mathbf{w}_{ij}(t+1) = \eta[(\sum_{k}^{N_i} \delta_{kj} \otimes \bar{\mathbf{w}}_{kj}) \odot \mathbf{F}'(\mathbf{net}_{ij})] \otimes \bar{\mathbf{o}}_i + \alpha \mathbf{w}_{ij}(t) \tag{3.29}$$

para cualquier salida k con una función de activación no lineal, se tiene

$$\mathbf{w}_{jk}(t+1) = \eta[(\mathbf{y}_{k_t} - \mathbf{y}_{k_a} \odot \mathbf{F}'(\mathbf{net}_{jk})] \otimes \bar{\mathbf{o}}_j + \alpha \mathbf{w}_{jk}(t) \tag{3.30}$$

y para cualquier salida k con una función de activación lineal

$$\mathbf{w}_{jk}(t+1) = \eta(\mathbf{y}_{k_t} - \mathbf{y}_{k_a}) \otimes \bar{\mathbf{o}}_j + \alpha \mathbf{w}_{jk}(t) \tag{3.31}$$

En estas ecuaciones, $\mathbf{F}$ es la función de activación definida de acuerdo a la ecuación 3.14, t es el paso de actualización, η y α son la "razón de aprendizaje" y el momento, respectivamente, $\otimes$ representa el producto geométrico, $\odot$ es el producto escalar, y $\bar{(\cdot)}$ es la "anti-involución multivectorial" (reversión o conjugación). En el caso del AG no Euclidiana $G_{0,3,0}$, $\bar{(\cdot)}$ corresponde a la simple conjugación. Cada neurona ahora consiste de $p + q + r$ unidades, cada una para un componente del multivector. Los bias son también multivectores y son absorbidos como se acostumbra en la suma de la señal de activación. En las reglas de aprendizaje, las ecuaciones (3.29 - 3.31), el cálculo del producto geométrico y la anti-involución varía dependiendo del AG que esté siendo usada [44]. Entonces, se tiene que la conjugación requerida en la regla de aprendizaje para el álgebra de cuaterniones es $\bar{\mathbf{x}} = x_0 - x_1\sigma_1 - x_2\sigma_2 - x_3\sigma_1\sigma_2$, donde $\mathbf{x} \in G_{0,2,0}$.

3.5.2. Redes neuronales recurrentes geométricas

No existen trabajos para diseñar redes recurrentes geométricas. Sólo se han extendido modelos de redes neuronales recurrentes valuadas con reales mediante la introducción del AG, se han propuesto así redes completamente conectadas que son extensiones de modelos recurrentes tipo Hopfield [20]. Estas extensiones del tipo Hopfield son una generalización de las redes clásicas recurrentes.

Capítulo 4

DISEÑO DE REDES NEURONALES GEOMÉTRICAS

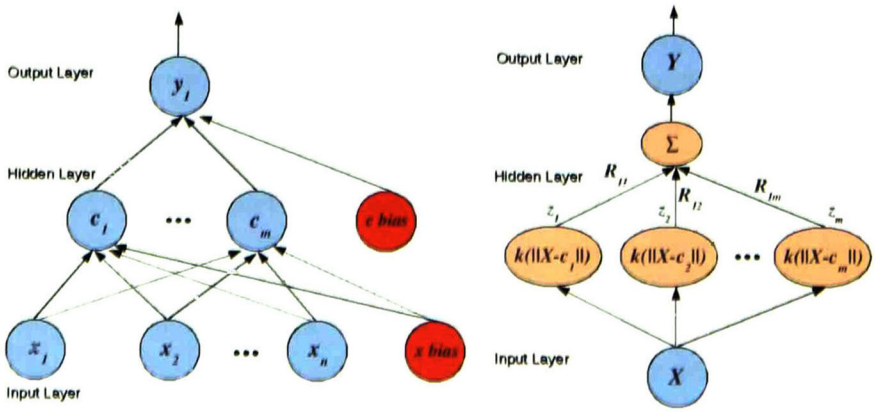
Un problema general en la definición de algoritmos implica definir el tipo de aplicación que tienen. En nuestro caso, trabajamos con algoritmos cuya aplicación permita ayudar a entender y manejar más fácilmente información relacionada con objetos geométricos y sus transformaciones. En el marco del AG esto es más sencillo.

Desde el punto de vista de aplicaciones de neurocomputación, es importante definir un modelo que permita trabajar con valores hipercomplejos y esto es posible usando multivectores del AG. Este tipo de modelos puede ser usado para encontrar transformaciones entre entidades expresadas en cuatro dimensiones o mayores, y como se mostrará, esto puede ser hecho únicamente en el marco del AG. Ya se ha dicho que las redes neuronales valuadas con reales, complejos y cuaterniones son simplemente casos particulares de las redes neuronales multidimensionales definidas con el AG. A continuación se explican nuestros modelos propuestos de RNAs donde las entradas, las salidas, pesos y neuronas ocultas son expresados usando valores con $k-$vectores. Nuestro modelo obtiene la transformación existente entre los $k-$vectores en la entrada y salida a través de la combinación de operadores geométricos, definida con AG, que nos permite tener una descripción más natural e intuitiva de la transformación implicada. Además, lo que es muy importante, se podrá leer de los

pesos de las RNAs geométricas, la transformación existente entre los datos de entrada y salida.

4.1. Aproximación de transformaciones

Cuando la función que se desea aproximar resulta en una transformación geométrica $T : \mathbb{M} \to \mathbb{M}$ donde $\mathbb{T}, \mathbb{M} \in \mathcal{G}_{p,q,r}$ entonces es factible un algoritmo que nos ayude a encontrar $\mathbb{T}$. Contamos con datos expresados en el lenguaje geométrico que permite el AG. Estos datos podemos usarlos para determinar las entradas $\mathbb{I}$ y salidas $\mathbb{O}$ de nuestro algoritmo. También $\mathbb{I}, \mathbb{O} \in \mathcal{G}_{p,q,r}$. La naturaleza de las variables y parámetros de nuestro modelo está determinada también en el marco de trabajo del AG. En la primera versión de nuestro modelo de red neuronal, se usan "rotores" como operadores geométricos definidos en $\mathcal{G}_{3,0,0}$ (Sección 2.4.1).



(a) Modelo valuado con reales (b) Modelo según ecuación 4.5

Figura 4.1: RBF Networks

4.2. Descripción de la red geométrica

La primera versión de nuestro modelo geométrico de RBF-N valuada con multivectores (GRBF-N), usa rotores como sus pesos. Puede ser usada en tiempo real para estimar transformaciones lineales cambiantes en espacio 3D que existan entre conjuntos de entidades geométricas. Para su diseño, usamos un modelo neuronal de RBF-N porque es adecuado para modelar mapeos no lineales y puede aprender rápidamente.

Las redes neuronales modulares, como estructuras combinadas, tienen también una base biológica: los sistemas neuronales naturales son compuestos de una jerarquía de redes construídas de elementos espacializados para tareas diferentes. En general, las redes combinadas son más poderosas que las de arquitectura plana no estructuradas.

El diseño de una red neuronal supervisada puede ser logrado de maneras diferentes. Puede ser visto como un problema de ajuste de curvas (aproximación) en un espacio de altas dimensiones. Así, el aprendizaje es equivalente a encontrar una superficie en un espacio multidimensional que proporcione un mejor ajuste a los datos de entrenamiento. Sus unidades ocultas proporcionan un conjunto de "funciones" (funciones de base radial) que constituyen una base arbitraria para los patrones de entrada cuando ellos son expandidos en el espacio oculto. La RBF-N tiene una arquitectura de "alimentación hacia adelante" e implica tres capas con roles totalmente diferentes. La capa de entrada es hecha de nodos fuente (unidades sensoriales) que conectan la red a su ambiente. La segunda capa, la única capa oculta en la red, aplica una transformación no lineal desde el espacio de entrada al espacio oculto; en la mayoría de las aplicaciones el espacio oculto es de alta dimensionalidad. La capa de salida es lineal, dando la respuesta de la red al patrón de activación aplicado a la capa de entrada. Una justificación matemática para la razón fundamental de la transformación no lineal seguida de una transformación lineal puede ser explicada como sigue: un problema de clasificación de patrones definido en un espacio de altas dimensiones es más probable a ser separable linealmente que en un espacio de bajas dimensiones, ésta es la razón para frecuentemente hacer mayor la dimensión del espacio oculto en la red. Otro punto importante es que

la dimensión del espacio oculto está directamente relacionado a la capacidad de la red para aproximar un mapeo suave de entrada-salida; entre más alta sea la dimensión del espacio oculto, más precisa será la aproximación [45]. Obviamente el límite está dictado por el parámetro que evita el sobre entrenamiento de la red y por ende la pérdida de la generalización.

4.2.1. El modelo de la RBF-N

Una variante de las redes híbridas, es decir, que tienen una capa Kohonen o de agrupación, consiste en usar Gausianas como función de activación de las unidades. La capa oculta es entrenada en la manera usual, pero la entrada es procesada diferentemente en cada unidad oculta. Éstas últimas producen un valor de salida que es combinado por el asociador lineal en la salida. Es deseable que la unidad oculta cuyo vector de peso ("centroide") que se encuentra más cerca al vector de entrada, se dispare más fuertemente que las otras unidades ocultas. Cada unidad oculta calcula su salida usando el vector de entrada, los centros vectoriales de las unidades ocultas y un valor que representa la distancia entre el centro vectorial $\mathbf{c}_i$ y su vecino más cercano. Los pesos de la capa final son determinados usando retropropagación. El error cuadrático está dado por:

$$E = \frac{1}{2}\left(\sum_{i=1}^{n} g_i(\mathbf{x})w_i - f(\mathbf{x})\right)^2 \tag{4.1}$$

Las actualizaciones necesarias de pesos son dadas por

$$\Delta w_i = -\frac{dE}{dw_i} = \gamma g_i(\mathbf{x})(f(\mathbf{x}) - \sum_{i}^{n} g_i(\mathbf{x})w_i) \tag{4.2}$$

La mezcla de gaussianas proporciona una aproximación continua de la función objetivo y hace innecesario el cálculo de la excitación máxima (saturación) en la capa oculta. El error puede ser minimizado usando más unidades ocultas. La principal diferencia entre redes hechas de funciones de base radial y las redes de unidades sigmoidales es que las primeras usan funciones concentradas localmente como bloques de construcción mientras que las últimas usan escalones suaves. Si la función a ser aproximada es una Gausiana, se necesita

organizar varios escalones sigmoidales para delimitar la región de interés. El tipo de función de activación a usar depende del problema a resolver.

La capa oculta de la RBF-N tiene m unidades ajustadas localmente, y son interconectadas a la capa de salida de L unidades lineales. Cada conexión es ponderada por un valor real. Todas las unidades ocultas reciben el vector $\mathbf{X}$ n-dimensional de entrada valuado con reales (Figure 4.1(a)). Cada salida de la unidad oculta z_j representa la cercanía de la entrada X al vector $n-$dimensional de parámetros c_j asociado con la j-ésima unidad oculta. La respuesta de la j-ésima unidad oculta ($j = 1, 2, ..., m$) se calcula como,

$$z_j = k\left(\frac{\|X - c_j\|}{\sigma^2{}_j}\right) \tag{4.3}$$

donde $k(\cdot)$ es una función positiva radialmente simétrica (*kernel*) con un máximo único en su "centro" c_j, la cual cae rápidamente a cero fuera del centro y tiene un valor apreciable únicamente cuando la distancia $\|X - c_j\|$ es más pequeña que σ_j. El parámetro σ_j es el ancho del campo receptivo en el espacio de entrada en la unidad j. Dado un vector de entrada X, la salida de RBF-N es el vector de actividad L-dimensional Y, cuyo l-ésimo componente ($l = 1, 2, \ldots L$) está dado por,

$$Y_l(X) = \sum_{m=1}^{M} w_{lj} z_j(X) \tag{4.4}$$

donde w_{lj} son los pesos de la red. La precisión de estas redes es controlada por tres parámetros: el número de funciones base, su localización y su ancho. Estos valores y los pesos aplicados a las salidas de las funciones RBF son determinados por un proceso de entrenamiento. El método de agrupación *K-Means* [45] es uno de varios métodos para encontrar los centros de grupo de las funciones RBF. Para encontrar los pesos de la capa de salida es conveniente usar un método de gradiente descendente.

La creación de redes neuronales en el marco del AG permite tener una perspectiva diferente de computación neuronal. El estudio se dirige al procesamiento de datos expresados de manera geométrica directamente como entradas de las redes neuronales que internamente realizan un procesamiento en términos de operadores geométricos. Nuestro trabajo está relacionado con la

visión computacional y robótica. Se usa un sistema estéreo para obtener información del medio ambiente y a partir de ésta construir las entidades usando AG.

Hemos considerado la arquitectura de una red modular para diseñar nuestra red neuronal geométrica. Consta de dos bloques geométricalmente definidos con la ayuda del AG. Hemos acoplado uno de auto-organización con otro de alimentación hacia adelante. Se aplica una versión del algoritmo de *K-Means* que trabaja con elementos geométricos para determinar los elementos representativos en el espacio de entrada. Cada unidad oculta calcula su salida usando el elemento de entrada que está definido por un multivector. El multivector puede representar un plano, una línea, un punto o algún otro elemento definido en AG. Evidentemente, en un principio, las unidades ocultas han sido definidas por elementos de la misma naturaleza pero de manera aleatoria.

Una vez definidos los elementos geométricos ocultos, entonces procede la determinación del segundo bloque de la red. Este bloque básicamente consiste en determinar la transformación geométrica para operar cada elemento oculto a partir de la similitud del elemento de entrada con alguno de ellos. Se considera la función del error cuadrático para comparar los elementos geométricos de salida resultantes de la red contra los que se consideran deseables. Para definir tal transformación geométrica se ha diseñado un versión del algoritmo *LMS* que trabaja con operadores y elementos definidos en el AG.

4.2.2. Redes de alimentación hacia adelante usando rotores

Es posible diseñar un modelo que permita encontrar una transformación, definida en términos geométricos, existente entre dos entidades definidas también geométricamente. El modelo permite determinar la transformación únicamente en términos de rotación usando multivectores. Es un modelo que permite encontrar tal transformación de manera adaptiva, lo cual es diferente a usar un conjunto de datos predeterminados para entrenar la GRBF-N usando rotores. Note que en las RNA's tradicionales no es posible, a partir de los pesos, obtener información como puede ser la transformación entre los datos

de entrada y salida.

GRBF-Ns usando rotores

Definamos nuestro esquema como $g(P, X)$ donde P es el conjunto de parámetros de nuestra red. X es la entrada y $g(\cdot)$ es la función determinada por la red (Figura 4.1(b)); la cual es definida por neuronas valuadas con multivectores que pueden expresar números hipercomplejos. Mediante un algoritmo de aprendizaje adaptivo, P es ajustado de manera que los datos de entrenamiento se adaptan al modelo de red lo mejor posible. $g(\cdot)$ usa funciones Gausianas base para definir la proximidad de X a los centros $\hat{\mathbf{c}}$. Nuestro esquema usa rotores ($\hat{\mathbf{R}}$) como pesos de la capa de salida que son combinados linealmente para definir la salida Y de la red. $X, Y, \hat{\mathbf{R}}, \hat{\mathbf{c}}$ pertenecen al AG $\mathcal{G}_3$ y son normalizados. Estos parámetros y los valores σ_i de cada centro, definen P.

Entrenamiento de GRBF-Ns usando rotores

El conjunto de entrenamiento es un par etiquetado X_i, Y_i que representa asociaciones de un mapeo dado. Dado el número de centros, el proceso de aprendizaje adaptivo inicia cuando el primer par de entrada de entrenamiento es presentado a la red. La GRBF-N usa el algoritmo de *K-Means*, adaptado a las necesidades geométricas, para determinar los centros de cada función Gausiana base y la distancia Euclidiana para indicar la proximidad de X_i a cada centro c_i. El parámetro σ_i es definido para cada unidad Gausiana y es igual a la distancia máxima entre los centros. De manera similar que en otros de nuestros modelos, para determinar $\hat{R}$ que mejor aproxime el mapeo entre X_i y Y_i, la GRFB-N usa un método de mínimo cuadrados ordinario (*LMS*) que puede trabajar con entidades y operadores geométricos para el entrenamiento. Nuestro esquema actualiza $\hat{R}$ considerando la rotación del centro c_i, esto se realiza usando el R_i asociado (peso) a c_i y la salida respectiva de la función de base radial. Dado que nuestro objetivo es analizar la factibilidad de nuestros diversos modelos, nuevamente se observa que el algoritmo de agrupación *K-Means* y el algoritmo *LMS* proceden con sus propios cálculos individuales de una manera concurrente, esto acelera el proceso de entrenamiento. Nuestro

modelo GRBF-N trabaja iterativamente como se describe en el Cuadro 4.1.

Es necesario indicar que hay que considerar un detalle importante justo después de actualizar el rotor $\mathbf{R}_i(n+1)$. El objetivo es ayudar a tener mejores aproximaciones de la transformación que se desea calcular. Se atiende el método descrito en [1]. El rotor estimado $\mathbf{r}_i^*$ consiste de dos vectores estimados de cuatro dimensiones $\mathbf{R}_i^*$ y $\mathbf{R}_i'^*$. Más adelante, en la versión de motores de nuestra red, se explica con más detalle este procedimiento de normalización.

Considerando la relación en la ecuación 4.8 (ver ecuación 2.58), las modificaciones pueden ser hechas simplemente considerando el rotor unitario en este caso. Como se observa, esta versión de GRBF-N usa el producto geométrico.

Podemos expresar ahora que la red es definida por,

$$g(P, X, n) = \sum_{i=1}^{M} R_i \left(z_i(n) c_i(n)\right) \widetilde{R}_i \tag{4.5}$$

4.2.3. Redes de alimentación hacia adelante usando motores

Existe una actualización de la la versión descrita anteriormente de nuestra GRBF-N, esta vez los parámetros son definidos usando multivectores de mayor dimensión en el marco matemático del AG.

Representación de líneas usando álgebra de motores

Las líneas pueden ser modeladas en un espacio 4D usando el álgebra especial de los motores $\mathcal{G}^+_{3,0,1}$, la cual extiende en 4D el espacio de líneas usando bases bivectoriales. Dado que esta álgebra es extendida únicamente por bivectores y escalares, se observa que esta AG especial es el sistema más apropiado para el modelado de líneas. Dado que el producto del seudoescalar unitario $I = e_1e_2e_3e_4$ con cualesquier bivector dual construido a partir de la base $\{e_4e_1, e_4e_2, e_4e_3\}$ es cero, se debe seleccionar la base bivectorial $\{e_2e_3, e_3e_1, e_1e_2\}$ para representar una línea como sigue:

$$\mathbf{L} = \mathbf{n} + I\mathbf{m} \tag{4.6}$$

Cuadro 4.1: El método de entrenamiento de una GRBF-N usando rotores

Algoritmo Entrena_GRBF-N($I, Y, \epsilon, \rho, \eta$)

1. *Inicialización.* Selecciona valores aleatorios para los centros iniciales $c_k(0)$, distribución uniforme en el intervalo [-10,10]; la única restricción es que estos valores iniciales sean diferentes. Puede ser también deseable mantener pequeña la norma Euclidiana de los centros.
2. Iterar hasta que no se observen cambios notables en los centros c_k o el error $< \epsilon$
   a) *Muestreo.* Extraer una entrada muestra **x**, que es expresada por un multivector en el espacio de entrada I. **x** es ingresado al algoritmo en la iteración n
   b) *Definición de similaridad.* $k(x)$ denota el índice del centro multivectorial que mejor se asemeje a la entrada **x**. Encontrar $k(x)$ en la iteración n usando el criterio Euclidiano de mínima distancia:
   $k(x) = \underset{k}{\text{argmin}} \|x(n) - c_k(n)\|, k = 1, 2, ..., m,$
   donde $c_k(n)$ es el centro de la k-ésima función radial en la n-ésima iteración. Se observa que la dimensión del k-vector x es la misma que la del k-vector c_k y en este caso es 4.
   c) *Actualización.* Ajuste de los centros de las funciones de base radial:
   $$c_k(n+1) = \begin{cases} c_k(n) + \rho[x(n) - c_k(n)], & \text{if } k = k(x), \\ c_k(n), & \textit{de otra manera} \end{cases}$$
   ρ es un *parámetro de razón de aprendizaje* en el rango $0 < \rho < 1$
   d) *Error.*
   $e(n) = Y(n) - \sum_{i=1}^{m} R_i(n)\,(z_i(n)c_i(n))\,\widetilde{R}_i(n)$
   e) *Actualización del vector de rotores.*
   $R_i(n+1) = R_i(n) + \eta z_i(n) e(n)$
   η es un *parámetro de razón de aprendizaje* en el rango $0 < \eta < 1$
   f) *Normalización del rotor.*
   Ver Ecuaciones 4.8, 4.9, 4.10, 4.11, 4.12, 4.13, 4.14, 4.15
   g) *Continuación.* Incrementar n con 1

donde $z_i(n) \in \mathbb{R}$ y $R_i(n), c_i(n), X_i(n), Y_i(n) \in \mathcal{G}_3$.

donde **n** es la dirección de la línea y **m** es el momento.

En este caso, los bivectores para la dirección de la línea y el momento son calculados usando dos puntos expresados con bivectores x_1 y x_2, que se encuentran sobre la línea, como sigue:

$$\begin{aligned}
\mathbf{n} &= (\mathbf{x_2} - \mathbf{x_1}) \\
&= (x_{21} - x_{11})e_2e_3 + (x_{22} - x_{12})e_3e_1 + (x_{23} - x_{13})e_1e_2 \\
&= L_{n1}e_2e_3 + L_{n2}e_3e_1 + L_{n3}e_1e_2 \\
\mathbf{m} &= \mathbf{x_1} \times \mathbf{x_2} \\
&= (x_{12}x_{23} - x_{13}x_{22})e_2e_3 + (x_{13}x_{21} - x_{11}x_{23})e_3e_1 + \cdots + (x_{11}x_{22} - x_{12}x_{21})e_1e_2 \\
&= L_{m1}e_2e_3 + L_{m2}e_3e_1 + L_{m3}e_1e_2
\end{aligned} \tag{4.7}$$

Esta representación de líneas usando números duales es fácil de entender y manipular algebraicamente, y es completamente equivalente a la representación en términos de coordenadas de Plücker. Usando la notación de corchetes, la ecuación de la línea llega a ser $\mathbf{L} \equiv (0, \mathbf{n}) + \mathbf{I}(0, \mathbf{m})$, donde **n** y **m** son extendidas con bases bivectoriales 3D. Los motores están expresados en el AG $\mathcal{G}_{3,0,1}$ (Sección 2.4.2).

El AG, comparada con el cálculo vectorial es más fácil e intuitiva debido a sus propiedades algebraicas y su representación más simple en rotaciones y translaciones que únicamente necesitan el eje de rotación "l", el ángulo "θ", y el vector de translación "**t**", respectivamente.

GRBF-Ns usando motores

La actualización de nuestro modelo implica considerar la translación y rotación de manera compacta [46]. Ahora se usan motores para rotar y transladar *blades* de cualquier grado usando ecuaciones simples. Esto es, no sólo vectores sino también líneas, planos, y cualesquier otro objeto geométrico que puede ser representado por un *blade* puede ser rotado y transladado con el motor que es estimado por nuestra red. Este esquema usa motores ($\hat{\mathbf{M}}$) como pesos en la capa de salida, los cuales son combinados linealmente para determinar la salida Y de la red. X, Y, $\hat{\mathbf{M}}$, $\hat{\mathbf{c}}$ pertenecen a $\mathcal{G}^+_{3,0,1}$.

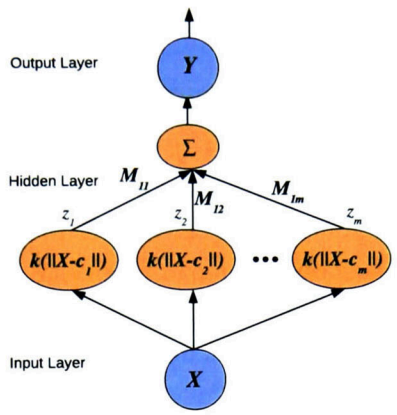


Figura 4.2: GRBF-N valuada con motores

Entrenamiento de GRBF-Ns usando motores

De manera similar al entrenamiento de la red definida con rotores, en este caso, el conjunto de entrenamiento es un par etiquetado X_i, Y_i que representa asociaciones de un mapeo dado que puede ser una transformación geométrica. Dada cierta cantidad de centros, el proceso de aprendizaje adaptivo inicia cuando el primer par de entrada de entrenamiento es presentado a la red. La GRBF-N que usa pesos valuados con motores, también usa el algoritmo de agrupación de *K-Means* para determinar los centros de cada función Gausiana de base y una distancia Euclidiana para determinar la proximidad de X_i a cada centro c_i. El parámetro σ_i es definido de igual manera que para el caso de la red con rotores. Para determinar $\hat{\mathbf{M}}$ que mejor aproxime los datos de muestra X_i, esta red GRFB-N también usa un método ordinario de mínimos cuadrados (LMS) con ciertas modificaciones requeridas por el contexto geométrico. Nuestro esquema actualiza el $\hat{M}$, considerando rotar y transladar el centro c_i usando el M_i asociado y la salida de la función RBF respectiva. El algoritmo de agrupación *K-Means* y el de LMS proceden concurrentemente con sus propios cálculos individuales como en la versión previa. La GRBF-N actualizada trabaja iterativamente como se indica en el Cuadro 4.2.

Aqui hay un detalle a considerar importante justo después de actualizar el motor $\mathbf{M}_i(n+1)$. El objetivo es ayudar a tener mejores aproximaciones de

Cuadro 4.2: El método de entrenamiento de una GRBF-N usando motores

Algoritmo Entrena_GRBF-N($I, Y, \epsilon, \rho, \alpha, \beta$)

1. *Inicialización.* Selecciona valores aleatorios para los centros iniciales $c_k(0)$.
2. Iterar hasta que no se observen cambios notables en los centros c_k o el error $< \epsilon$
   a) *Muestreo.* Extraer una entrada muestra **x**, que es expresada por un multivector en el espacio de entrada I. **x** es ingresado al algoritmo en la iteración n.
   b) *Definición de similaridad.* $k(x)$ denota el índice del centro multivectorial que mejor se asimila a la entrada **x**. Encontrar $k(x)$ en la iteración n usando el criterio Euclidiano de mínima distancia:
   $k(x) = \underset{k}{\text{argmin}} ||x(n) - c_k(n)||, k = 1, 2, ..., m,$
   $c_k(n)$ es el centro de la k-ésima función radial en la iteración n.
   La dimensión del k-vector x es la misma que la del k-vector c_k y en este caso es 6 porque lo usamos para determinar una línea.
   c) *Actualización.* Ajuste de los centros de las funciones de base radial:
   $$c_k(n+1) = \begin{cases} c_k(n) + \rho[x(n) - c_k(n)], & \text{if } k = k(x), \\ c_k(n), & \textit{de otra manera} \end{cases}$$
   ρ es un *parámetro de razón de aprendizaje* en el rango $0 < \rho < 1$
   d) *Error.*
   $e(n) = Y(n) - \sum_{i=1}^{m} \mathbf{M}_i(n) c_i(n) \widetilde{\mathbf{M}}_i(n)$
   e) *Actualización del vector de motores.*
   $ETemp = z_i(n) e(n) X(n)$
   $\mathbf{R}_i(n) = extractRotor(M_i(n))$
   $\mathbf{T}_i(n) = extractTraslator(M_i(n))$
   $\mathbf{R}_i(n+1) = \mathbf{R}_i(n) + \alpha ETemp_{RotorPart}$
   (*Normalización del rotor.*
   Ver Ecuaciones 4.8, 4.9, 4.10, 4.11, 4.12, 4.13, 4.14, 4.15)
   $\mathbf{T}_i(n+1) = \mathbf{T}_i(n) + \beta ETemp_{TraslatorPart}$
   $\mathbf{M}_i(n+1) = \mathbf{T}_i(n+1)\mathbf{R}_i(n+1)$
   $i = 1, ...M$
   α, β son *parámetros de razón de aprendizaje* en el rango $0 < \eta, \beta < 1$
   f) *Continuación.* Incrementar n con 1

donde $\alpha, \beta, z_i(n) \in \mathbb{R}$ y $M_i(n), c_i(n), X_i(n), Y_i(n) \in \mathcal{G}^+_{3,0,1}$.

la transformación que se desea calcular. Se sigue el método descrito en [1]. El motor estimado $\mathbf{M}_i^*$ consiste de dos vectores estimados de cuatro dimensiones $\mathbf{R}_i^*$ y $\mathbf{R}_i'^*$

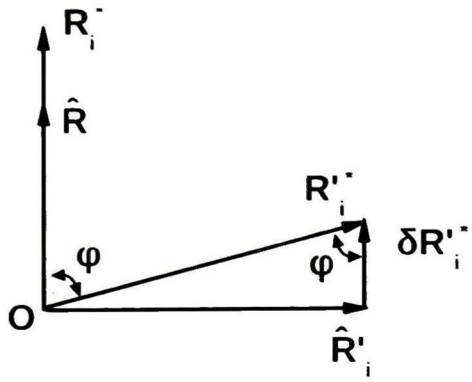


Figura 4.3: Restricción de ortogonalidad de acuerdo a la Ecuación 4.10

$$\mathbf{R}^T\mathbf{R} = 1 \tag{4.8}$$

Considerando la relación en la ecuación 4.8, las modificaciones pueden ser hechas simplemente considerando el rotor unitario en este caso.

$$\hat{\mathbf{R}}_i = \frac{\mathbf{R}_i^*}{\|\mathbf{R}_i^*\|} \tag{4.9}$$

Sin embargo, la restricción

$$\mathbf{R}'^T\mathbf{R} = 0 \tag{4.10}$$

no es tan simple de satisfacer. Ésta nos indica que $\mathbf{R}$ debe ser ortogonal al rotor dual $\mathbf{R}'$. Pero en la práctica, el rotor estimado $\mathbf{R}_i^*$ usualmente no es ortogonal al rotor dual estimado $\mathbf{R}_i'^*$. La figura 4.3 sugiere como hacer cumplir esta restricción geométrica para modificar la orientación de $\mathbf{R}_i'^*$. Para realizar esto, se considera el coseno del ángulo φ entre las estimaciones $\mathbf{R}_i^*$ y $\mathbf{R}_i'^*$:

$$cos(\varphi) = \frac{\mathbf{R}_i'^{*T}\mathbf{R}_i^*}{\|\mathbf{R}_i'^*\| \cdot \|\mathbf{R}_i^*\|} \tag{4.11}$$

Esta ecuación es simplificada usando la ecuación 4.9 y el rotor unitario $\hat{\mathbf{R}}_i$ como se indica

$$cos(\varphi) = \frac{\mathbf{R}_i'^{*T}\hat{\mathbf{R}}}{\|\mathbf{R}_i'^*\|} \tag{4.12}$$

Posteriormente, consideramos la derivación a partir del rotor ortogonal ideal $\hat{\mathbf{R}}'$,

$$\delta\hat{\mathbf{R}}_i'^{*} = \|\hat{\mathbf{R}}_i'^{*}\| cos(\varphi)\hat{\mathbf{R}}_i = (\mathbf{R}_i'^{*T}\hat{\mathbf{R}}_i)\mathbf{R}_i \tag{4.13}$$

Finalmente, se calcula simplemente el rotor ortogonal ideal $\hat{\mathbf{R}}_i'$, primero se calcula el rotor unitario $\mathbf{R}_u$ ortogonal a $\hat{\mathbf{R}}_i$:

$$\mathbf{R}_u = \frac{(\mathbf{R}_i'^{*} - \delta\hat{\mathbf{R}}_i'^{*})}{\|\mathbf{R}_i'^{*} - \delta\hat{\mathbf{R}}_i'^{*}\|} \tag{4.14}$$

entonces se multiplica por el valor absoluto del rotor estimado $\mathbf{R}_i'^{*}$

$$\hat{\mathbf{R}}_i' = \|\mathbf{R}_i'^{*}\|\mathbf{R}_u \tag{4.15}$$

Hemos rotado el vector estimado $\mathbf{R}_i'^{*}$ hasta que sea ortogonal a $\hat{\mathbf{R}}_i$.

Este ajuste también fue considerado en esta versión de la GRBF-N con el objetivo de tener mejores aproximaciones de la transformación calculada. Como podemos observar, esta versión de la GRBF-N también usa el producto geométrico. Note que $g(P, X, n) = \sum_{i=1}^{m} M_i(n)z_i(n)c_i(n)\widetilde{M}_i(n)$

4.3. Redes recurrentes valuadas con multivectores

Uno de los motivos principales para diseñar este tipo de redes neuronales, es hacer posible el aprendizaje basado en la experiencia considerando cambios en el tiempo en un contexto geométrico. Una red estática puede ser extendida incluyendo ciclos de retro-alimentación. A diferencia de las redes de alimentación hacia adelante, donde hay una relación algebraica entre la entrada y salida, la arquitectura recurrente contiene memoria, es decir, es un sistema dinámico. La red recurrente contiene la red hacia adelante como un caso especial y obviamente representa una clase más general de arquitecturas.

La manera quizás más fácil de incorporar información secuencial o temporal en una situación de entrenamiento es crear el dominio temporal espacial y usar una arquitectura de retro-alimentación hacia adelante. La información disponible que está de regreso en el tiempo es insertada ampliando el espacio de entrada de acuerdo al tamaño de la "ventana" fija y predeterminada

$\mathbf{X} = \mathbf{x(t), x(t-1), x(t-2), \ldots, x(t-\omega)}$. Esto frecuentemente es llamado línea de retraso de reuso dado que las entradas son puestas en un contenedor retardado y discretamente desplazado conforme el tiempo pasa.

Los modelos de redes recurrentes que se han diseñado en esta investigación determinan sus parámetros usando multivectores. Se establecen ciertos bucles de retro-alimentación que son representados mediante multivectores. Estos bucles pueden ser de tipo global o local. Se puede tener retroalimentación desde las neuronas de salida de la red neuronal a la capa de entrada o desde las neuronas ocultas de la red a la capa de entrada (retroalimentación global). Estos modelos puedan ser usados como memorias asociativas o redes de mapeo entrada-salida. Nuestros modelos básicamente establecen una red de mapeo entrada-salida que permita identificar la transformación geométrica variante en el tiempo. La red recurrente geométrica responde "temporalmente" a una señal de entrada (determinada con AG) aplicada de manera externa.

Esto es útil para el procesamiento de secuencias de objetos geométricos. Se desea que las neuronas ocultas definan el "estado" de la red. El comportamiento dinámico de una red recurrente clásica en general lo definimos como

$$\mathbf{x}(n+1) = \mathbf{f}(\mathbf{x}(n), \mathbf{u}(n)) \tag{4.16}$$

$$\mathbf{y}(n) = \mathbf{C}\mathbf{x}(n) \tag{4.17}$$

donde $\mathbf{f}(\cdot,\cdot)$ es una función no lineal caracterizando la capa oculta, y $\mathbf{C}$ es la matriz de los pesos caracterizando la capa de salida. Se han considerado aspectos importantes del modelo de la "red recurrente simple" descrita por Elman, sin embargo, hemos creado nuestros modelos recurrentes actualizando nuestra GRBF-N valuada con motores (ver Figura 4.4).

4.3.1. Modelos de redes recurrentes geométricas

Nuestros modelos de redes neuronales recurrentes se han implementado utilizando álgebra geométrica para diseñar la arquitectura de la red recurrente creada a partir de nuestra GRBF-N valuada con motores; denominamos a nuestro modelo general como "Red Recurrente Geométrica de Funciones de Base

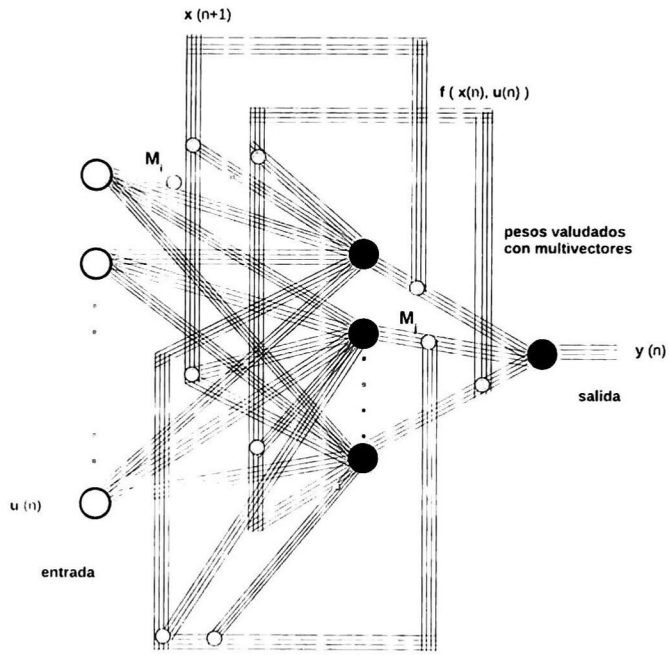


Figura 4.4: Red recurrente geométrica simple

Radial (Recurrent Geometric Radial Basis Function-Network)" (RGRBF-N). Los pesos en estos modelos están definidos con multivectores, específicamente con motores. La diferencia entre los modelos que se presentan a continuación, se encuentra en la conexión para lograr la recurrencia. Creamos la recurrencia en nuestras redes simplemente retroalimentando la salida de ciertas neuronas hacia la entrada de otras, ya sea de la misma o diferente capa. La forma de interconexión entre estas neuronas generan diferentes efectos en las salidas de las redes recurrentes y esto permite que sean útiles para diferentes tareas. Los modelos de RGRBF-N usan una representación interna del tiempo. Esta propiedad se obtiene ya sea mediante la autoconexión de las neuronas en la capa de entrada o conectando las salidas de las neuronas de la capa de entrada u oculta hacia neuronas de otras capas en la red.

La RGRBF-N Tipo Elman

Este modelo contiene conexiones recurrentes de las neuronas ocultas a una capa de "unidades de contexto" consistente de retrasos. Estas unidades de contexto almacenan las salidas de las neuronas ocultas para un paso de tiempo, y posteriormente son alimentadas nuevamente a la capa de entrada. Esto permite que las neuronas ocultas tengan algún registro de sus activaciones previas y hace posible que la red realice tareas de aprendizaje que se extienden a lo largo del tiempo. Las neuronas ocultas también alimentan a las neuronas de salida que reportan la respuesta de la red al estímulo aplicado externamente. Debido a la naturaleza de la retroalimentación alrededor de las neuronas ocultas, ellas pueden continuar reciclando la información a través de la red sobre múltiples pasos temporales, y por lo tanto descubrir representaciones abstractas de tiempo. Basados en el modelo de Elman, pretendemos reconocer el movimiento de cierto objeto geométrico (una línea) en un flujo continuo de orientaciones diferentes con la restricción de que el movimiento sea suave (Figura 4.5). Este modelo se puede visualizar en la figura 4.6(c), donde se aprecia que la entrada a la red representa a la línea en su orientación actual. La salida representa la mejor estimación de la red para indicar la siguiente orientación de la línea en la secuencia. El rol de las unidades de contexto es proveer a la red de una "memoria dinámica" con el fin de codificar la informa-

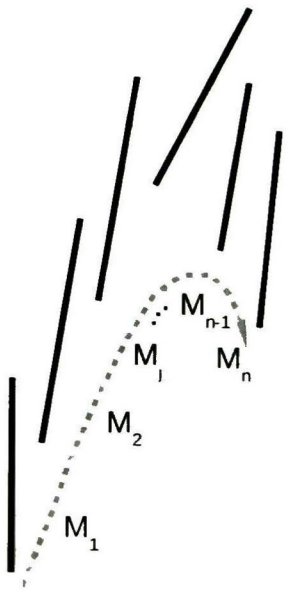


Figura 4.5: Transformación geométrica variante en el tiempo de un objeto en movimiento. M_i indica transformaciones rígidas en términos de motores.

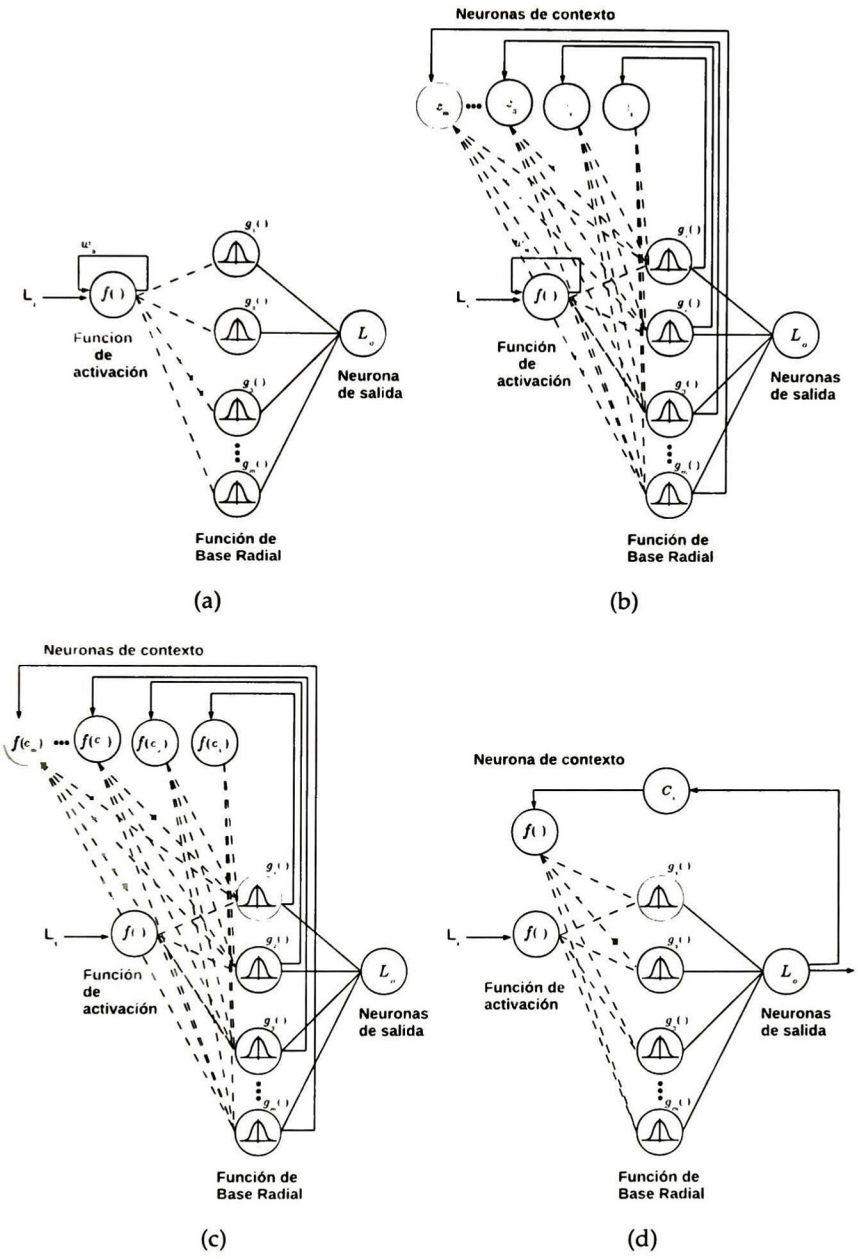


Figura 4.6: Redes geométricas recurrentes RBF. SISO. Las entradas y salidas son líneas L_i y L_o

análisis experimental de nuestro trabajo y finalmente el capítulo seis expresa las conclusiones y trabajo futuro de nuestra investigación.

Capítulo 2

COMPUTACIÓN GEOMÉTRICA

La geometría sintética, geometría de coordenadas, números complejos, cuaterniones, análisis vectorial y de tensores, álgebra de matrices, álgebra de Grassmann, álgebra de Clifford, álgebra de *Spinors*, cinemática, espacio de Plücker y geometría proyectiva poseen conceptos geométricos. Además, existen consecuencias innecesarias de muchos lenguajes como el aprendizaje redundante, complejidad de acceso al conocimiento, traducción frecuente, y más. Esto es una buena razón para considerar un lenguaje que involucre las técnicas anteriores para resolver problemas relacionados..

El álgebra geométrica (AG) es un lenguaje para geometría que explota el concepto de un vector. Define "multivectores" mediante la combinación de representaciones con dimensiones diferentes: escalares, vectores, bi-vectores, trivectores y k-vectores. Usa dimensiones de las representaciones llamadas *grades*. Es adecuada para representar estructuras y desarrollar algoritmos que se apliquen en ingeniería. Ofrece las herramientas necesarias y de unificación para expresar geometría y sus relaciones sin la necesidad de cambiar de sistema matemático o hacer excepciones en casos especiales, esto permite que la comunicación de las ideas sea más fácil, lo que da lugar a desarrollar algoritmos de manera rápida e intuitiva [21]. Ofrece el potencial para realizar optimizaciones e implementaciones altamente eficientes y se confía que será competitiva con métodos clásicos cuando también se adapten algoritmos a sus nuevas capacidades [22].

valores de las unidades ocultas en el tiempo t transformados por un motor fijo. Estas unidades de contexto por lo tanto proveen a la red de una memoria. La figura 4.6(c) es un ejemplo de una red recurrente simple geométrica, se trata de la RGRBF-N, basada en nuestro modelo GRBF-N. Está diseñada con el objetivo de que los nuevos modelos basados en este inicial, puedan aprender un número ilimitado de secuencias de longitud variable. Se pretende que las unidades de entrada y salida representen líneas individuales, con las que la red podría ser entrenada para predecir la próxima línea en una secuencia de líneas. La naturaleza de estas redes se comprende simplemente refiriendo a la orientación de la línea en orden serial (i.e.; $L_0, L_1, L_2, L_3, \ldots$) considerando la definición de la línea en $\mathcal{G}_{3,0,1}$ como lo indica la ecuación 4.18.

$$\begin{aligned} \mathbf{L} &= L_{n_1}e_2e_3 + L_{n_2}e_3e_1 + L_{n_3}e_1e_2 + L_{m_1}e_4e_1 + L_{m_2}e_4e_2 + L_{m_3}e_4e_3 \\ &= L_{n_1}e_2e_3 + L_{n_2}e_3e_1 + L_{n_3}e_1e_2 + I(L_{m_1}e_2e_3 + L_{m_2}e_3e_1 + L_{m_3}e_1e_2) \\ \mathbf{L}_i &= \mathbf{n}_i + \mathbf{Im}_i, \text{ donde } i = 1, 2, \ldots \end{aligned} \tag{4.18}$$

En la figura 4.5 se puede apreciar una secuencia de líneas. Cada secuencia inicia y termina con una misma línea terminal. Para cualquier otra línea que no sea la inicial o final, el orden de su orientación dentro de la secuencia debe ser coherente con el movimiento suave de la línea. La arquitectura de la red tiene una neurona de entrada y una neurona de salida, ambas representan líneas en diferentes orientación dentro de la secuencia (incluyendo la inicial y terminal). Se usa una cantidad variante de unidades ocultas (cantidad determinada a priori, de acuerdo al problema), dependiendo del tamaño de la secuencia, esa misma cantidad corresponde a unidades de contexto. En trabajos futuros se pretende que la cantidad de unidades varíe incrementalmente de manera automatizada durante la etapa de entrenamiento.

$$\mathbf{L}_0 = (1)*e_2e_3 + (0.1)*e_3e_1 + (0.1)*e_1e_2 + (0.1)*e_4e_1 + (0.1)*e_4e_2 + (0.1)*e_4e_3 \tag{4.19}$$

$$\begin{aligned}
\mathbf{L}_0 &= (1) * e_2e_3 + (0.1) * e_3e_1 + (0.1) * e_1e_2 + (0.1) * e_4e_1 + (0.1) * e_4e_2 + (0.1) * e_4e_3 \quad (4.20)\\
\mathbf{L}_1 &= (0.1) * e_2e_3 + (1) * e_3e_1 + (0.1) * e_1e_2 + (0.1) * e_4e_1 + (0.1) * e_4e_2 + (0.1) * e_4e_3\\
\mathbf{L}_2 &= (0.1) * e_2e_3 + (0.1) * e_3e_1 + (1) * e_1e_2 + (0.1) * e_4e_1 + (0.1) * e_4e_2 + (0.1) * e_4e_3\\
\mathbf{L}_3 &= (0.1) * e_2e_3 + (0.1) * e_3e_1 + (0.1) * e_1e_2 + (1) * e_4e_1 + (0.1) * e_4e_2 + (0.1) * e_4e_3\\
\mathbf{L}_4 &= (0.1) * e_2e_3 + (0.1) * e_3e_1 + (0.1) * e_1e_2 + (0.1) * e_4e_1 + (1) * e_4e_2 + (0.1) * e_4e_3\\
\mathbf{L}_5 &= (0.1) * e_2e_3 + (0.1) * e_3e_1 + (0.1) * e_1e_2 + (0.1) * e_4e_1 + (0.1) * e_4e_2 + (1) * e_4e_3
\end{aligned}$$

De este modo, podemos representar la línea terminal con 1's y 0's (valor muy pequeño). La ecuación 4.19 corresponde a la línea terminal. De manera que la secuencia podría ser dada por la ecuación 4.20. El entrenamiento de la red para una secuencia de líneas en particular, implica varios pasos, el número de éstos depende de la longitud de la secuencia. Al inicio del entrenamiento, las activaciones de las unidades de contexto se determinan como lo indica la ecuación 4.21. La línea terminal es primero presentada a las unidades de entrada y la red predice el sucesor. El error (la diferencia entre el sucesor predicho y el real especificado por la secuencia de entrenamiento) es determinado y considerado para el reajuste de los motores (pesos). Las unidades de contexto reciben una copia de las activaciones de las unidades ocultas, y la próxima línea en la secuencia de entrenamiento (la cuál fue el objetivo para la unidad de salida del primer paso del entrenamiento) es presentada a la unidad de entrada. El entrenamiento continua de este modo hasta que otra instancia de la línea terminal es alcanzada.

$$\mathbf{L}_i = 0.5 * e_2e_3 + 0.5 * e_3e_1 + 0.5 * e_1e_2 + 0.5 * e_4e_1 + 0.5 * e_4e_2 + 0.5 * e_4e_3 \quad (4.21)$$

De manera similar a los previos modelos, el conjunto de entrenamiento es una secuencia de líneas orientadas de manera diferente, representando un movimiento suave. X_i, Y_i representa el mapeo de un par de estas líneas dentro de la secuencia en un tiempo determinado. Dado el número de centros, el proceso de aprendizaje adaptivo inicia cuando el primer par de entrada de entrenamiento es presentado a la red. La RGRBF-N usa el algoritmo de *K-Means* para determinar los centros de cada función Gausiana base y la distancia Euclidiana para indicar la proximidad de X_i, y de cada salida L_i de las unidades de contexto, a cada centro c_i. Cada neurona oculta tiene una función

de activación asociada que genera una línea, la cual es transformada por un motor y nuevamente es comparada con los centros de la misma manera que las entradas a la red. Es aqui donde aparece la recurrencia. El parámetro σ_i es definido para cada unidad Gausiana y es igual a la distancia máxima entre los centros. Para determinar $\hat{M}$ que mejor aproxime el mapeo entre X_i y Y_i, la RGRFB-N usa un método de mínimos cuadrados ordinario (*LMS*) que puede trabajar con objetos y operadores geométricos para el entrenamiento. Nuestro esquema actualiza $\hat{M}$ considerando la rotación y traslación del centro c_i, esto se realiza usando el M_i asociado (peso) a c_i y la salida respectiva de la función de base radial. Como es posible observar, el algoritmo de agrupación *K-Means* y el algoritmo *LMS* proceden con sus propios cálculos individuales de una manera concurrente, lo cual acelera el proceso de entrenamiento. Nuestra RGRBF-N trabaja iterativamente como se muestra en el Cuadro 4.3.

Se aclara que en el algoritmo presentado en el Cuadro 4.3, se usa únicamente el rotor del motor, es decir, la traslación aun no se contempla, sin embargo se ha creado la estructura con motores considerando el trabajo a futuro de construir una arquitectura recurrente considerando la traslación y rotación al mismo tiempo mediante motores.

La RGRBF-N basada en la autoconexión

La autoconexión de neuronas en la capa de entrada da a la red geométrica recurrente un carácter dinámico con la simplicidad del proceso de entrenamiento (Figura 4.10(a)). Ésta autoconexión ha sido usada en un perceptrón multicapa (MLP) y en una red recurrente RBF valuada con reales [47, 2]. La mayor desventaja del MLP es la complejidad de su proceso de entrenamiento. En nuestro trabajo, el ajuste de parámetros es un tema importante y se enmarca en el contexto geométrico dado que la información de entrada está definida en AG. La flexibilidad del proceso de entrenamiento de la RGRBF-N representa una ventaja importante de nuestras arquitecturas.

Cuadro 4.3: El método de entrenamiento de una RGRBF-N tipo Elman

Algoritmo Entrena_RGRBF-N($I, Y, \epsilon, \rho, \eta$)

Para cada secuencia de entrenamiento, realizar los pasos del 1 al 2.

1.1 *Inicialización*. Selecciona valores aleatorios para los centros iniciales $c_k(0)$; la única restricción es que estos valores iniciales sean diferentes. Puede ser también deseable mantener pequeña la norma Euclidiana de los centros.

1.2 Determinar las activaciones de las unidades de contexto (Ecuación 4.21).

2. Iterar hasta que no se observen cambios notables en los centros c_k o el error $< \epsilon$ o encontrar la segunda instancia de la línea terminal.

a) *Muestreo*. Extraer una entrada muestra **x** (línea), que es expresada por un multivector en el espacio de entrada I. **x** es ingresado al algoritmo mediante la neurona de entrada en la iteración n.
La línea **l** resulta de aplicar el motor a la línea resultante de una neurona oculta en la iteración n. Se guarda en las unidades de contexto y se alimenta a la red.

b) Presentar el sucesor (línea) a la unidad de salida como respuesta deseada.

Calcular el sucesor predicho por la red recurrente

c) *Definición de similaridad*. $k(x)$ denota el índice del centro multivectorial que mejor se asemeje a la entrada **x**. Las líneas **l** de las neuronas recurrentes también se comparan a los centros. Encontrar $k(x)$ en la iteración n usando el criterio Euclidiano de mínima distancia:

$$k(x) = \underset{k}{\operatorname{argmin}} \|r(n) - c_k(n)\|, k = 1, 2, \ldots, m,$$

donde $c_k(n)$ es el centro de la k-ésima función radial en la n-ésima iteración. $r(n) \in \{x(n), l(n)\}$. Se observa que la dimensión del k-vector x es la misma que la del k-vector c_k y en este caso es 8.

d) *Actualización*. Ajuste de los centros de las funciones de base radial:

$$\mathbf{c}_k(n+1) = \begin{cases} c_k(n) + \rho[r(n) - c_k(n)], & \text{if } k = k(x), \\ c_k(n), & \textit{de otra manera} \end{cases}$$

ρ es un *parámetro de razón de aprendizaje* en el rango $0 < \rho < 1$.

Determinar el error y actualizar los motores (pesos)

e) *Error*.

$$e(n) = Y(n) - \sum_{i=1}^{m} M_i(n)\,(z_i(n)c_i(n))\,\widetilde{M}_i(n).$$

f) *Actualización del vector de motores*.

$$ETemp = z_i(n)e(n)X(n)$$
$$\mathbf{R}_i(n) = extractRotor(M_i(n))$$
$$\mathbf{T}_i(n) = extractTraslator(M_i(n))$$
$$\mathbf{R}_i(n+1) = \mathbf{R}_i(n) + \alpha ETemp_{RotorPart}$$
$$\mathbf{T}_i(n+1) = \mathbf{T}_i(n) + \beta ETemp_{TraslatorPart}$$
$$\mathbf{M}_i(n+1) = \mathbf{T}_i(n+1)\mathbf{R}_i(n+1)$$
$$i = 1, \ldots M$$

(*Normalización del rotor*. Ver Ecuaciones 4.8, 4.9, 4.10, 4.11, 4.12, 4.13, 4.14, 4.15)

α, β son *parámetros de razón de aprendizaje* en el rango $0 < \eta, \beta < 1$.

g) *Continuación*. Incrementar n con 1.

h) Probar la condición de parada:

Si la línea objetivo es igual a la segunda instancia de la línea terminal, entonces
Parar
De otro modo,
Copiar las activaciones de las unidades ocultas a las unidades de contexto y continuar con el paso a)

donde $z_i(n) \in \mathbb{M}$ y $M_i(n), l(n), c_i(n), X_i(n), Y_i(n) \in \mathcal{G}_3$.

El efecto de la autoconexión

Cada neurona de la capa de entrada realiza una suma en el instante **t** entre su entrada $\mathbf{L}_i$ y su salida del instante previo (**t**−1) que ha sido transformada por el motor (peso) de la autoconexión $\mathbf{w}_{ij}$. Su salida es el resultado de la función de activación:

$$\mathbf{a}_i(\mathbf{t}) = \mathbf{w}_{ii}\mathbf{x}_i(\mathbf{t}-1) + \mathbf{L}_i \tag{4.22}$$

$$\mathbf{x}_i(\mathbf{t}) = f(\mathbf{a}_i(\mathbf{t})) \tag{4.23}$$

donde $\mathbf{a}_i(\mathbf{t})$ y $\mathbf{x}_i(\mathbf{t})$ son respectivamente la activación y la salida de la neurona i en el tiempo **t**, f representa la función de activación de la neurona geométrica **i** que se basa en la función sigmoide. Y $\mathbf{w}_{ii}$ es el motor (peso) de la autoconexión de la neurona **i**. Para estudiar el efecto de la autoconexión de cada neurona, se considera la entrada $L_i = 0$ y la salida de la neurona $\mathbf{x}_i(\mathbf{t}) = 1$. La neurona evolucionará por lo tanto, sin la influencia de la entrada externa ($L_i = 0$) [47]. La evolución con respecto al tiempo de la neurona de salida está dada por la función multivectorial que se define con la función ():

$$\mathbf{x}_i(\mathbf{t}) = \frac{1 - expM(-k(w_{ii}x(t-1)\widetilde{w_{ii}}))}{1 + expM(-k(w_{ii}x(t-1)\widetilde{w_{ii}}))} \tag{4.24}$$

El diagrama de la figura 4.8 muestra la evolución de la salida de la neurona en

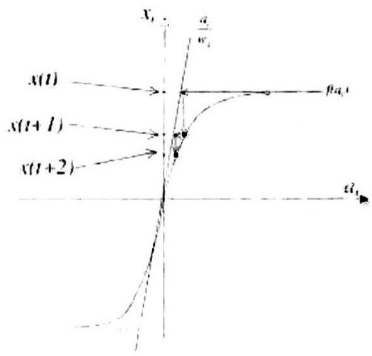


Figura 4.8: El efecto de la autoconexión en la evolución del estado neuronal [2]

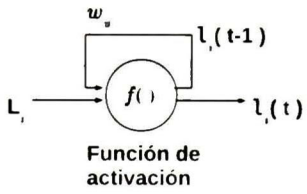


Figura 4.9: Neurona geométrica recurrente.

el tiempo considerando el caso valuado con números reales (Ecuación 4.25, la función *exp*() es valuada con reales). Esta evolución depende del gradiente de Δ y también del valor del parámetro k de la función de activación. La función de activación de las neuronas recurrentes (Figura 4.9) recibe un multivector que representa la suma de dos líneas $\mathbf{L}_a$ y $\mathbf{L}_b$ como lo indica la ecuación 4.26. Se obtiene una línea $\mathbf{L}_c$

$$\mathbf{x}_i(t) = \frac{1 - exp(-kw_{ii}x(t-1))}{1 + exp(-kw_{ii}x(t-1))} \tag{4.25}$$

$$\begin{aligned} \mathbf{f}(\mathbf{L}_c) &= \mathbf{f}(w_{ii}\mathbf{L_a} + \mathbf{L}_b) \qquad (4.26) \\ \mathbf{f}(\mathbf{n}_c + I\mathbf{m}_c) &= \mathbf{f}\left((\mathbf{n}_a + w_{ii}(I\mathbf{m}_a)) + (\mathbf{n}_b + I\mathbf{m}_b)\right) \\ \mathbf{f}(\mathbf{n}_c + I(\mathbf{t}_c \wedge \mathbf{n}_c)) &= \mathbf{f}\left((\mathbf{n}_a + w_{ii}I(\mathbf{t}_a \wedge \mathbf{n}_a)) + (\mathbf{n}_b + I(\mathbf{t}_b \wedge \mathbf{n}_b))\right) \end{aligned} \qquad (4.27)$$

Entonces, podremos calcular $\mathbf{n}_c$ y $\mathbf{m}_c$

$$\mathbf{n}_c = \frac{\mathbf{n}_a + \mathbf{n}_b}{|\mathbf{n}_a + \mathbf{n}_b|} \tag{4.28}$$

$$\mathbf{m}_c = \mathbf{t}_c \wedge \mathbf{n}_c \tag{4.29}$$

Pero para calcular $\mathbf{t}_c$, se considera

$$\begin{aligned} \mathbf{m}_c &= \mathbf{m}_a + \mathbf{m}_b \qquad (4.30) \\ \mathbf{t}_c \wedge \mathbf{n}_c &= w_{ii}(\mathbf{m}_a \cdot \mathbf{n}_a) + (\mathbf{m}_b \cdot \mathbf{n}_b) \\ \mathbf{t}_c \wedge \mathbf{n}_c &= w_{ii}(\mathbf{m}_a \cdot \mathbf{n}_a) + (\mathbf{m}_b \cdot \mathbf{n}_b) \\ \mathbf{t}_c \wedge \mathbf{n}_c \cdot \mathbf{n}_c &= w_{ii}((\mathbf{t}_a \wedge \mathbf{n}_a) \cdot \mathbf{n}_a) + ((\mathbf{t}_b \wedge \mathbf{n}_b) \cdot \mathbf{n}_b) \\ \mathbf{t}_c &= w_{ii}t_a + t_b \end{aligned}$$

El algoritmo de entrenamiento usado para este modelo de red RGRBF-N, es muy similar al algoritmo previamente presentado, también involucra conceptos geométricos al momento de ajustar los motores (pesos) de la red. Este algoritmo permite trabajar con elementos definidos en el lenguaje del AG. La única diferencia es que considera en cada tiempo de entrenamiento a la secuencia completa con el objetivo de asociarla a un elemento geométrico determinado.

Entrenamiento de la RGRBF-N basada en la autoconexión

Este modelo recurrente geométrico acepta la secuencia de líneas completa en cada instante de tiempo, por lo que el modelo es de tipo MIMO (Figura 4.10(c)). La red relaciona el conjunto de líneas y sus diversas instancias en diferentes instantes de tiempo, mediante la recurrencia entre las neuronas ocultas y de entrada en la red. Al final, la secuencia de líneas es asociada con un elemento geométrico que se encuentra en la salida. El modelo ha sido diseñado para aprender patrones variantes en el tiempo o secuenciales. Nuevamente trabajamos con secuencias de líneas. El algoritmo puede reconocer secuencias de líneas. Con ajustes adecuados, los algoritmos podrían ser útiles para cualquier problema que implique secuencias de elementos geométricos. Cada unidad de entrada recibe información de la orientación de una línea y se combina con la información de la salida de la función de activación de cada neurona autoconectada. La salida de la red es comparada con la salida deseada y se usa el error para incrementalmente ajustar los motores que determinan los pesos. Las conexiones recurrentes hacia la misma neurona determinan un motor (Ecuación 4.31) cuyos elementos están todos definidos como lo indica la Ecuación 4.32 y no son reajustables. La estrategia para determinar los valores de los coeficientes depende del problema y refiere a un trabajo experimental y empírico. En el próximo paso en el tiempo $t + 1$ la secuencia de líneas en la entrada evoluciona en el tiempo (es muy parecida a la del tiempo t). Esta vez las unidades de contexto son las mismas que las que reciben los datos de entrada. Estas unidades de contexto por lo tanto proveen a la red de una memoria. Se inicializan los coeficientes del motor con un valor igual para ejecutar el algoritmo de aprendizaje. Evidentemente, al finalizar el algoritmo, el motor tendrá diferentes valores en sus coeficientes.

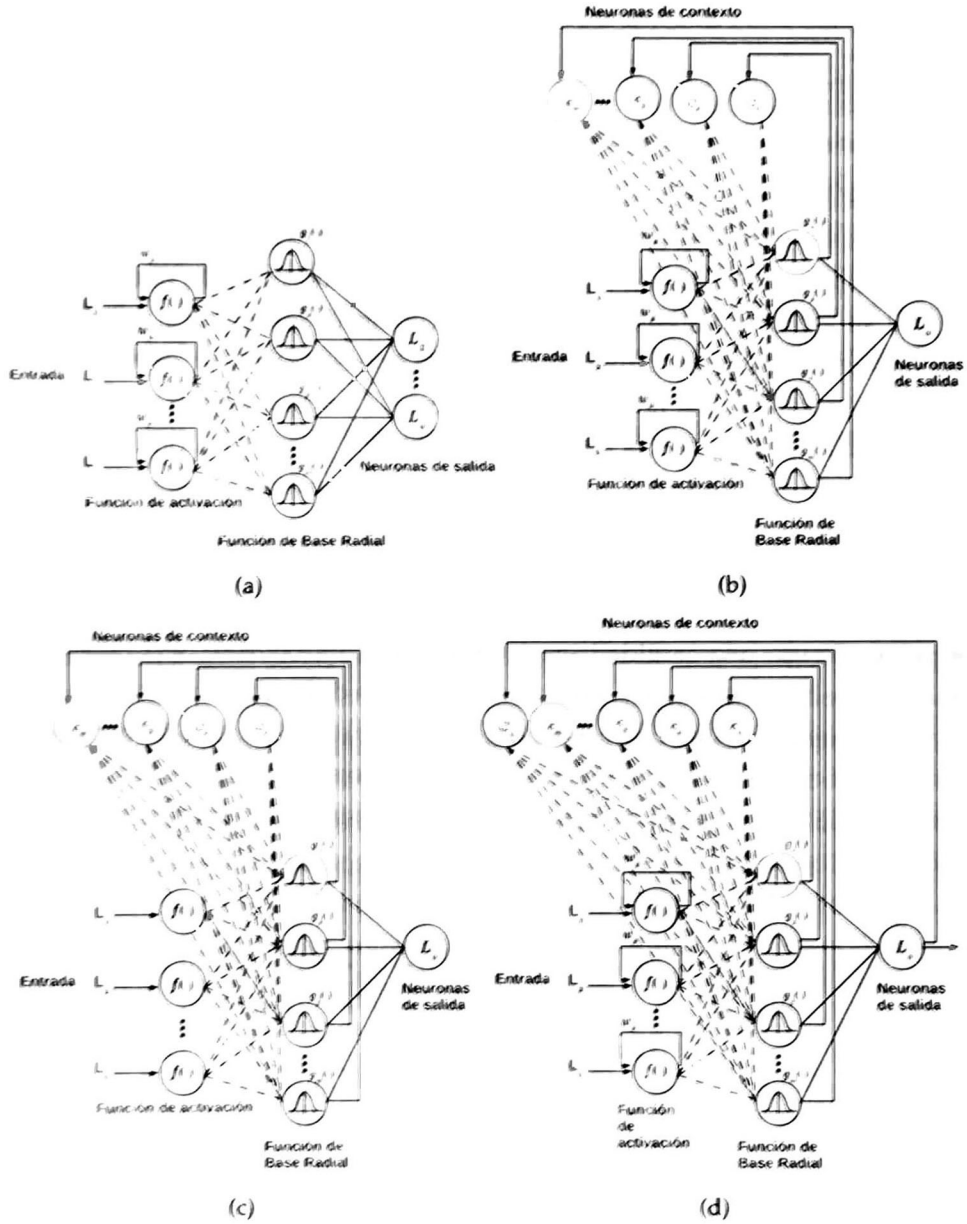


Figura 4.10: Redes geométricas recurrentes RBF. MIMO. Las entradas y salidas son líneas L_i y L_o

$$\mathbf{M} = \mathbf{T}_s\mathbf{R}_s = (a_0 + a_1 * e_2e_3 + a_2 * e_3e_2 + a_3 * e_2e_1) + I(b_0 + b_1 * e_2e_3 + b_2 * e_3e_2 + b_3 * e_2e_1) \quad (4.31)$$

$$\mathbf{M}_i = (0.8 + 0.8 * e_2e_3 + 0.8 * e_3e_2 + 0.8 * e_2e_1) + I(0.8 + 0.8 * e_2e_3 + 0.8 * e_3e_2 + 0.8 * e_2e_1) \quad (4.32)$$

La figura 4.10(c) muestra que el modelo está basado en la red GRBF-N. Está diseñada con el objetivo de que al aplicar mejoras al modelo, pueda aprender un número ilimitado de secuencias de longitud variable, donde cada secuencia se asocia a un elemento geométrico determinado en la salida de la red. Las entradas definen una secuencia de líneas y la salida el elemento geométrico asociado (una línea) a esa secuencia. Este tipo de red recurrente permite asociar una secuencia de líneas variante en el tiempo a otra entidad definida por un multivector (en este caso es otra línea). La arquitectura de la red tiene tantas neurona de entrada como líneas existen en una secuencia y una neurona de salida (para el elemento geométrico asociado). En nuestra investigación, las neuronas de entrada y salida representan líneas en diferentes orientación. Se usa una cantidad variante de unidades ocultas, dependiendo del tamaño de la secuencia. Las líneas se encuentran definidas también con 1's ó 0's (valor muy pequeño). La inicialización de las líneas se hace de manera que todos los coeficientes tienen el mismo valor, así entonces comienza el proceso de aprendizaje. De manera que la secuencia puede ser el patrón que indica la Ecuación 4.33.

$$\begin{aligned}
\mathbf{L}_0 &= (1) * e_2e_3 + (0.1) * e_3e_1 + (0.1) * e_1e_2 + (0.1) * e_4e_1 + (0.1) * e_4e_2 + (0.1) * e_4e_3 \quad (4.33)\\
\mathbf{L}_1 &= (0.1) * e_2e_3 + (1) * e_3e_1 + (0.1) * e_1e_2 + (0.1) * e_4e_1 + (0.1) * e_4e_2 + (0.1) * e_4e_3\\
\mathbf{L}_2 &= (0.1) * e_2e_3 + (0.1) * e_3e_1 + (1) * e_1e_2 + (0.1) * e_4e_1 + (0.1) * e_4e_2 + (0.1) * e_4e_3\\
\mathbf{L}_3 &= (0.1) * e_2e_3 + (0.1) * e_3e_1 + (0.1) * e_1e_2 + (1) * e_4e_1 + (0.1) * e_4e_2 + (0.1) * e_4e_3\\
\mathbf{L}_4 &= (0.1) * e_2e_3 + (0.1) * e_3e_1 + (0.1) * e_1e_2 + (0.1) * e_4e_1 + (1) * e_4e_2 + (0.1) * e_4e_3\\
\mathbf{L}_5 &= (0.1) * e_2e_3 + (0.1) * e_3e_1 + (0.1) * e_1e_2 + (0.1) * e_4e_1 + (0.1) * e_4e_2 + (1) * e_4e_3
\end{aligned}$$

Al inicio del entrenamiento, las activaciones de las unidades de contexto se determinan como la Ecuación 4.34 lo indica. El valor de los coeficientes se determina de acuerdo al problema. En este caso se usa un valor específico de acuerdo a un análisis experimental y empírico. Esta estrategia consiste en determinar los valores de los coeficientes de los motores de la red geométrica,

de acuerdo al problema. Básicamente, tales valores se ajustan considerando los resultados del algoritmo dependiendo del problema específico. Para todos nuestros modelos, un trabajo a futuro será aplicar técnicas de ajustes a estos valores, que sean inherentes al proceso de entrenamiento. La red entrenada podrá asociar a la secuencia de líneas en un tiempo t con otra línea dado que la red recibe en el entrenamiento a la misma secuencia de líneas pero en diferentes tiempos de manera que las autoconexiones son las que permiten mantener en memoria las secuencias previas. El error (la diferencia entre la salida de la red y el valor deseado) es determinado y considerado para el reajuste de los motores (pesos). Las unidades que reciben los datos de entrada reciben los resultados (líneas) de sus propias funciones de activación transformadas mediante el motor fijo w_{ii} determinado por la autoconexión de las neuronas de entrada.

$$\mathbf{L}_a = (0.5)*e_2e_3+(0.5)*e_3e_1+(0.5)*e_1e_2+(0.5)*e_4e_1+(0.5)*e_4e_2+(0.5)*e_4e_3 \quad (4.34)$$

El conjunto de entrenamiento es un par etiquetado S_i, Y_i que representa asociaciones de un mapeo dado, entre una secuencia de líneas S_i (determinada por líneas) y una línea que se encuentra en la neurona de salida. Dado el número de centros, el proceso de aprendizaje adaptivo inicia cuando el primer par de entrada de entrenamiento es presentado a la red. La RGRBF-N usa el algoritmo de *K-Means* para determinar los centros de cada función gausiana base y la distancia Euclidiana para indicar la proximidad de la línea l_j, de la secuencia S_i, a cada centro c_i. Las neuronas de entrada reciben los datos externos (líneas) y las salidas de sus funciones de activación mediante sus autoconexiones (Ecuación 4.26). El parámetro σ_i es definido para cada unidad Gausiana y es igual a la distancia máxima entre los centros. Con el objetivo de determinar $\hat{M}$ que mejor aproxime el mapeo entre los elementos de S_i y Y_i, nuestro modelo de RGRBF-N usa un método de mínimos cuadrados ordinario (*LMS*) que puede trabajar con objetos y operadores geométricos para el entrenamiento. Este esquema actualiza $\hat{M}$ considerando la rotación y traslación del centro c_i, esto se realiza usando el M_i asociado (peso) a c_i y la salida respectiva de la función de base radial. Se observa que el algoritmo de agrupación *K-Means* y el algoritmo *LMS* proceden con sus propios cálculos individuales de una manera concurrente, lo cual acelera el proceso de entrenamiento. Nuestro modelo de RGRBF-N trabaja iterativamente como se describe en el Cuadro 4.4.

Cuadro 4.4: El método de entrenamiento de una RGRBF-N basada en la autoconexión

Algoritmo Entrena_RGRBF-N($I, Y, \epsilon, \rho, \eta$)

1.1 *Inicialización.* Selecciona valores aleatorios para los centros iniciales $c_k(0)$; la única restricción es que estos valores iniciales sean diferentes. Puede ser también deseable mantener pequeña la norma Euclidiana de los centros.

1.2 Determinar las activaciones de las unidades de contexto como lo indica la ecuación 4.34.

2. Iterar hasta que no se observen cambios notables en los centros c_k o el error $< \epsilon$ o encontrar la segunda instancia de la línea terminal.

a) *Muestreo.* Determinar la muestra de la secuencia de entrada **S** (conjunto de líneas), que es expresada por un conjunto de multivectores en el espacio de entrada I. **S** es ingresada al algoritmo mediante las neuronas de entrada (líneas) en la iteración n

b) Determinar la salida de las funciones de activación de cada neurona considerando las autoconexiones y las líneas que determinan **S**. Tales salidas $l(n)$ se alimentan hacia la estructura de la red.

c) *Definición de similaridad.* $k(x)$ denota el índice del centro multivectorial que mejor se asemeje a la entrada $l(n)$. Encontrar $k(x)$ en la iteración n usando el criterio Euclidiano de mínima distancia:

$$k(x) = \underset{k}{\operatorname{argmin}} \|l(n) - c_k(n)\|, k = 1, 2, ..., m,$$

donde $c_k(n)$ es el centro de la k-ésima función radial en la n-ésima iteración. Se observa que la dimensión del k-vector $l(n)$ es la misma que la del k-vector c_k y en este caso es 6.

d) *Actualización.* Ajuste de los centros de las funciones de base radial:

$$c_k(n+1) = \begin{cases} c_k(n) + \rho[l(n) - c_k(n)], & \text{if } k = k(x), \\ c_k(n), & \textit{de otra manera} \end{cases}$$

ρ es un *parámetro de razón de aprendizaje* en el rango $0 < \rho < 1$

Determinar el error y actualizar los motores (pesos)

d) *Error.*

$$e(n) = Y(n) - \sum_{i=1}^{m} M_i(n)\,(z_i(n)c_i(n))\,\widetilde{M}_i(n)$$

e) *Actualización del vector de motores.*

$$ETemp = z_i(n)e(n)X(n)$$
$$\mathbf{R}_i(n) = extractRotor(M_i(n))$$
$$\mathbf{T}_i(n) = extractTraslator(M_i(n))$$
$$\mathbf{R}_i(n+1) = \mathbf{R}_i(n) + \alpha ETemp_{RotorPart}$$
$$\mathbf{T}_i(n+1) = \mathbf{T}_i(n) + \beta ETemp_{TraslatorPart}$$
$$\mathbf{M}_i(n+1) = \mathbf{T}_i(n+1)\mathbf{R}_i(n+1)$$
$$i = 1, ...M$$

(*Normalización del rotor.* Ver Ecuaciones 4.8, 4.9, 4.10, 4.11, 4.12, 4.13, 4.14, 4.15)

α, β son *parámetros de razón de aprendizaje* en el rango $0 < \eta, \beta < 1$

f) *Continuación.* Incrementar n con 1

donde $z_i(n) \in \mathbb{M}$ y $M_i(n), c_i(n), L_i(n), Y_i(n) \in \mathcal{G}_3$.

Capítulo 5

ANÁLISIS EXPERIMENTAL

Mucho del poder de esta álgebra radica en la manera en que es capaz de tratar con rotaciones mediante elementos llamados "rotores" En robótica se trabaja mucho con movimientos de cuerpos rígidos ligados, y es claro entonces que una buena descripción de rotaciones, translaciones y rotaciones relativas es esencial. Además de que un AG de 3D da muchas ventajas sobre otros sistemas, las rotaciones y translaciones son operaciones fundamentalmente diferentes en el álgebra. La primera versión de nuestra red neuronal trabaja usando rotores como sus pesos. Los experimentos se han realizado usando secuencias de imágenes obtenidas con un sistema estéreo. Podemos observar la evolución del aprendizaje que realiza nuestra primera versión de la GRBF-N. Usamos las orientaciones de conjuntos de líneas y planos. Este algoritmo es un enfoque prometedor para aplicaciones de neurocomputación donde se requieran considerar aspectos geométricos.

Los modelos de redes neuronales artificiales, como una extensión natural, están basadas en las propiedades de aproximación de funciones de las redes neuronales estáticas y son limitadas por el tiempo de pertenencia a un conjunto cerrado. Los pesos de la red neuronal son adaptados en línea para minimizar el error de identificación. Recordemos que las redes neuronales pueden ser estáticas (*feedforward*) o dinámicas (recurrentes o diferenciales). La mayoría de las redes neuronales estáticas son implementadas para la aproximación de funciones no lineales. La principal desventaja de estas redes es que

la actualización de los pesos no utiliza alguna información sobre la estructura temporal de datos locales y la aproximación de la función es sensible a los datos de entrenamiento. Las redes dinámicas pueden satisfactoriamente superar esta desventaja asi como presentar comportamiento adecuado en la presencia de dinámicas no modeladas, porque su estructura incorpora retroalimentación. Tienen capacidades de representación poderosas. Se usan redes neuronales dinámicas multicapa que contienen adicionales capas ocultas para mejorar las capacidades de aproximación; son como perceptrones multicapa combinados con operadores dinámicos. Cuando una red neuronal RBF es usada para clasificación, se resuelve este problema transformandolo dentro de un espacio de alta dimensión. La justificación para hacerlo está dado por el toerema de Cover [28], que establece que un problema de clasificación es más probable a ser separable linealmente en un espacio de alta dimensión que en uno de baja dimensión.

La otra justificación teórica es la teoría de regularización para la solución de problemas mal planteados (*ill posed problems*). Para los problemas de aproximación, la idea básica es estabilizar la solución por medio de un auxiliar no negativo funcional que incorpora información previa, y por lo tanto transforma un problema mal planteado en uno bien plateado.

Un enfoque comun para codificar información temporal usando redes neuronales estáticas es incluir entradas y salidas de retardo. Pero esta representación es limitada, dado que puede únicamente codificar un número finito de previas salidas medidas y entradas impuestas; por otra parte, tiende a requerir prohibitivamente largas cantidades de memoria, lo que dificulta su uso para casi todos los sistemas dinámicos de orden relativamente bajo. Como una alternativa prometedora y eficiente, la comunidad de investigación internacional ha estado explorando el uso de redes neuronales dinámicas y recurrentes. Éstas últimas se distinguen de las estáticas en que tienen al menos un ciclo de retroalimentación. Los bucles de retroalimentación implican el uso, para tiempo discreto, de ramificaciones compuestas de elementos de retraso unitarios denotados por q^{-1}, tal que: $u(k-1) = q^{-1}u(k)$, con k indicando la muestra k-ésima en el tiempo. Los bucles de retroalimentación resultan en un comportamiento dinámico no lineal debido a la función de activación no lineal

de las neuronas. Así que el término de red neuronal dinámica describe mejor esta estructura de redes neuronales: las redes neuronales dinámicas. Este tipo de redes permite un mejor entendimiento de estructuras biológicas y pueden ofrecer también grandes ventajas computacionales. Una arquitectura estática y una dinámica son equivalentes, pero desde el punto de vista computacional, un sistema con retroalimentación es equivalente a una estructura de datos larga, posiblemente infinita y estática.

5.1. La GRBF-N construida con rotores

Debido al ruido en visión robótica, nos concentramos en experimentos con líneas y planos observados para hacer un seguimiento de la relación espacial entre estas entidades geométricas. En esta primera versión de la red neuronal, nos limitaremos a considerar únicamente la transformación de rotación. Los próximos experimentos muestran que trabajando con GRBF-Ns, es posible detectar rotaciones entre líneas y entre planos. En esta propuesta se usan datos capturados con un sistema de visión estereo. Cada orientación de línea puede ser representada usando un bivector y esto también es verdad para la orientación del plano. Se consideran dos puntos, $\mathbf{p}$ and $\mathbf{q}$, para calcular la dirección n. Es posible usar nuestra GRBF-N para encontrar la rotación entre dos orientaciones de líneas y dos orientaciones de planos (Figura 5.4). Tenemos el plan de que mediante entrenamiento simultáneo, construimos un conjunto de dos GRBF-Ns como se muestra en la figura 5.4. Los pares de entrada (para entrenamiento y prueba) de orientaciones de líneas y planos fueron generados usando un disco con marcas distribuidas uniformemente (con distancias pequeñas entre ellos en una zona de 2 cm^2), las cuales representan un punto de los dos necesitados para construir la línea mientras que el otro punto (origen) fue fijado. Mediante el cambio de un punto, generamos un conjunto de orientaciones que están en diferente posición en el espacio en 3D (Figura 5.1). De este modo tenemos un conjunto de orientaciones de líneas. De manera similar, generamos un conjunto de orientaciones de plano. Estas orientaciones constituyen el conjunto de entrenamiento de ambas redes. Deseamos mostrar que nuestro esquema es un bloque de cómputo que puede ser usado en un

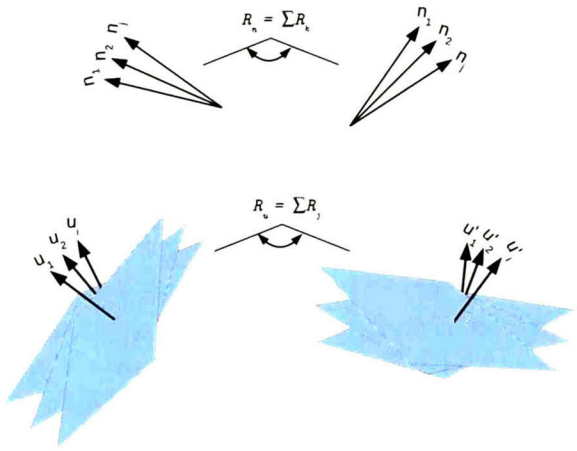


Figura 5.1: Conjuntos de orientaciones de líneas y planos.

sistema que puede ayudarnos a determinar otro tipo de transformaciones más complicadas (Figura 5.9). Para generar los conjuntos de prueba se procedió de manera similar. Como lo muestra la figura 5.2, la red GRBF-N aproxima el mapeo entre $\boldsymbol{n}_i$ y $\boldsymbol{n}'_i$. Usamos estos conjuntos de datos para entrenar y probar la GRBF-N. La figura 5.2 muestra el error entre el rotor calculado y el verdadero. Aunque no contamos hasta el momento con un análisis de convergencia, nuestros experimentos prácticos nos indican que esto es verdad ya que se alcanza la orientación objetivo de manera satisfactoria (Figura 5.7). Esto muestra que el rotor converge al valor esperado. Un resultado similar puede ser observado en la figura 5.3 para la aproximación del mapeo entre $\boldsymbol{u}_i$ y $\boldsymbol{u}'_i$ usando una segunda GRBF-N. Note que en estos experimentos estamos usando bivectores para expresar la dirección de la línea $\boldsymbol{n}_i$ y la orientación del plano $\boldsymbol{u}_i$. También, los rotores (que son los pesos del modelo) están expresados con bivectores. Usamos los valores $\eta = 0.9$, $\rho = 0.05$ en la primera red y $\eta = 0.6$, $\rho = 0.1$ en la segunda; en ambos casos fue suficiente usar tres centroides (neuronas ocultas), una neurona de entrada y una más en la salida. Las figuras 5.2 y 5.3 muestran que la GRBF-N es capaz de aproximar la transformación requerida para lograr la orientación objetivo a partir de una orientación inicial. A través de ellas, se puede observar que al principio, la red desconoce completamente la transformación entre la orientación de entrada y la orientación de salida.

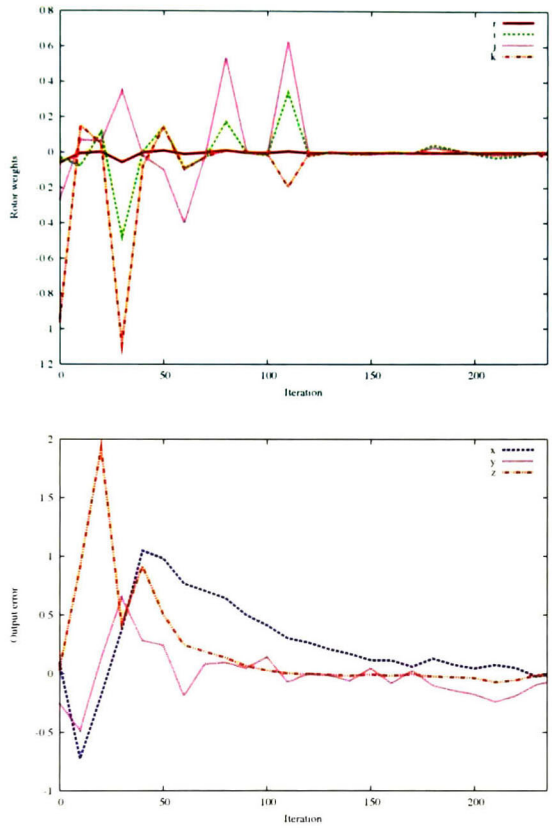


Figura 5.2: Aproximación de la transformación entre las orientaciones de las líneas $\mathbf{n_i}$ y $\mathbf{n'_i}$. Evolución del vector de rotores (arriba). Error en la salida durante el proceso de aprendizaje (abajo). Las salidas son expresadas como $X = xe_1e_2 + ye_2e_3 + ze_3e_1$

Sin embargo, mientras más pares de orientaciones son alimentados a la red, ésta puede aproximar de manera más precisa la rotación verdadera entre las entidades geométricas que se mueven suavemente. En robótica, el grado de precisión que se tiene es suficientemente útil para guiar a un robot. Hemos mostrado como la GRBF-N puede ayudar a aproximar la verdadera transformación entre ciertas entidades geométricas. Nuestro esquema converge de tal manera que usando diferentes orientaciones de entrenamiento y prueba, obtenemos un error de 0.0740 en la primera red y 0.0627 en la segunda. Este

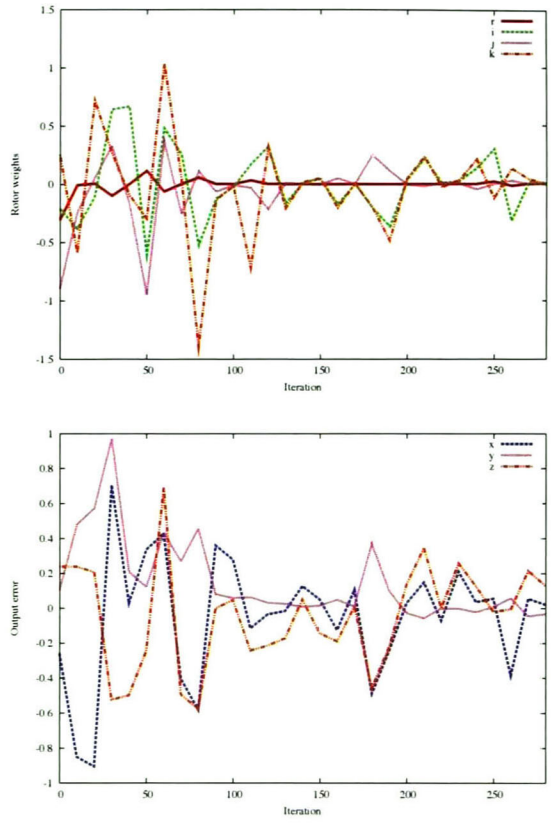


Figura 5.3: Aproximación de la transformación entre las orientaciones de planos $\mathbf{u}_i$ y $\mathbf{u'}_i$. Error en la salida durante el proceso de aprendizaje (abajo). Las salidas son expresadas como $X = xe_1e_2 + ye_2e_3 + ze_3e_1$

error es definido usando un promedio de los errores entre la distancia Euclidiana entre las orientaciones objetivo y las orientaciones calculadas por la GRBF-N. Esto indica que nuestro esquema trabaja bien.

En otro experimento, nuevamente obtuvimos datos a partir de un sistema de visión estéreo (Figura 5.5). Tenemos objetos cuyo eje de rotación está marcado por dos puntos coloreados bien distruibuidos. Con técnicas de visión, estos dos puntos son identificados en espacio 3D y entonces se calcula la orientación del objeto. Tenemos que I es la orientación de un objeto y O la orientación de otro objeto (o el mismo) y son expresadas usando multivectors de cuatro

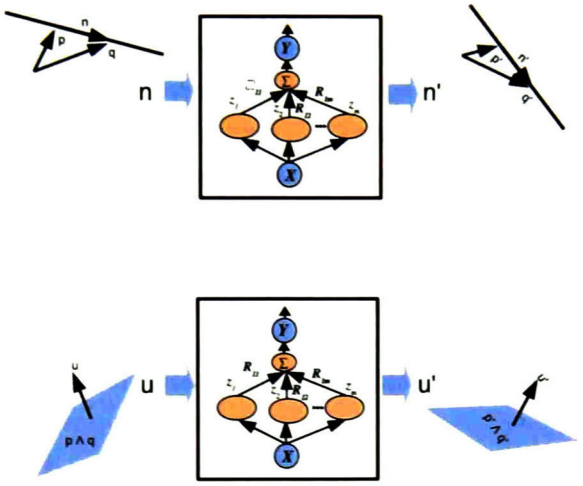


Figura 5.4: El conjunto de GRBF-Ns.

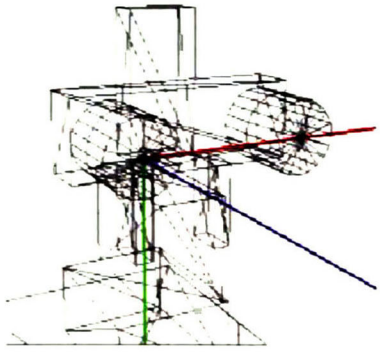


Figura 5.5: Sistema de visión estéreo.

dimensiones. Entonces se construyen pares de entrada (para el entrenamiento y prueba) únicamente perturbando con ruido Gaussiano la orientación I para determinar X_i y de la misma manera afectando O para determinar Y_i (Figura 5.9). Usamos estos conjuntos de datos para entrenar y probar la GRBF-N. La figura 5.6 (arriba) muestra que los valores de los rotores convergen a un valor esperado y esto también se observa en la figura 5.6 (abajo) donde se muestra que las salidas de la GRBF-N están muy cerca de los valores deseados. En estos experimentos estamos usando multivectores de cuatro dimensiones para expresar las orientaciones de los objetos (planos) y rotores. Usamos los valores $\eta = 0.1$ $\rho = 0.3$, tres centroides fueron suficientes, una entrada y una salida. La

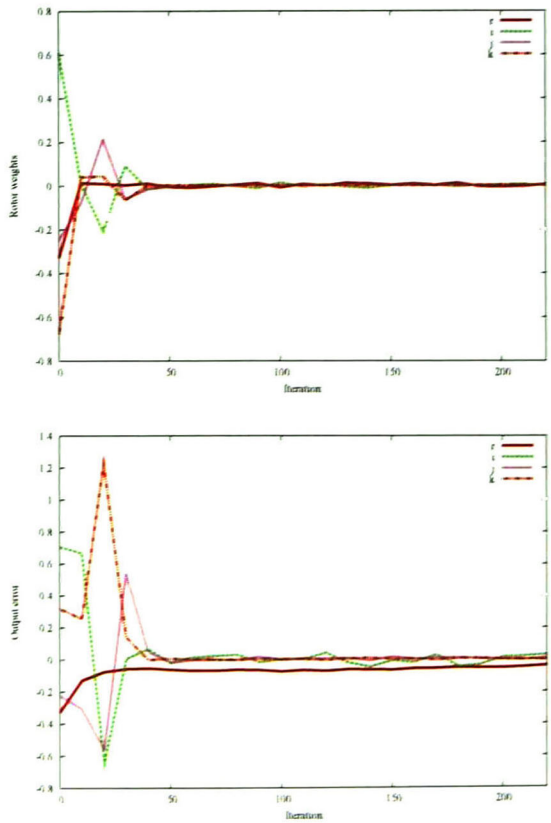


Figura 5.6: Evolución del rotor (arriba). Error en la salida durante el proceso de aprendizaje (abajo).

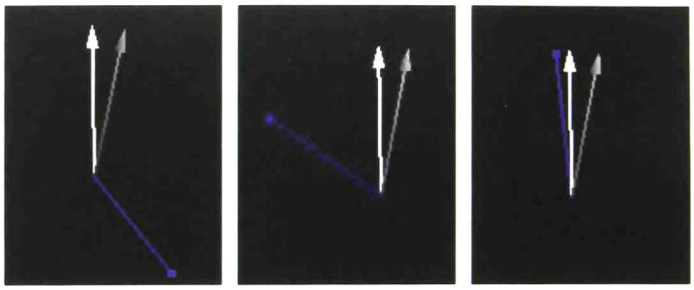
figura 5.7 muestra como nuestra red estima la transformación necesitada para lograr la orientación objetivo a partir de una inicial, en estas figuras podemos apreciar ambas orientaciones. Nuevamente, se observa que al inicio, la salida de la red está lejos de la orientación deseada. Conforme la GRBF-N aprende, su salida es más cercana al objetivo deseado. También, nuestro esquema converge en este experimento de manera que usando diferentes muestras de los datos de prueba se obtiene un error de 0.039, el cual está definido usando un promedio de la distancia Euclidiana entre las orientaciones objetivo y las salidas de la GRBF-N que también son orientaciones. También observamos que nuestro esquema trabaja bien en este experimento. La figura 5.9 ilustra que la GRBF-N es útil para definir un movimiento más complicado construyendo un conjunto de bloques para aprender una transformación específica de manera que la GRBF-N i aprende la transformación i (que puede ser expresada por un motor M, un rotor T o un traslador T).

El AG ayuda aqui también para expresar las entidades geométricas como círculos, esferas, planos, etc., de tal manera que podamos tener transformaciones entre orientaciones de diferentes tipos de entidades geométricas expresadas usando multivectores. Esto puede ser usado para indicar a un robot cuáles son las transformaciones necesarias, para mover partes de sus brazos y piernas con base en datos geométricos obtenidos con un sistema de visión. Usando un equipo de cómputo poderoso y cámaras externas al robot es posible lograr esta estructura y el robot conocerá en tiempo real la transformación a aplicar para mover una parte de su brazos o pierna.

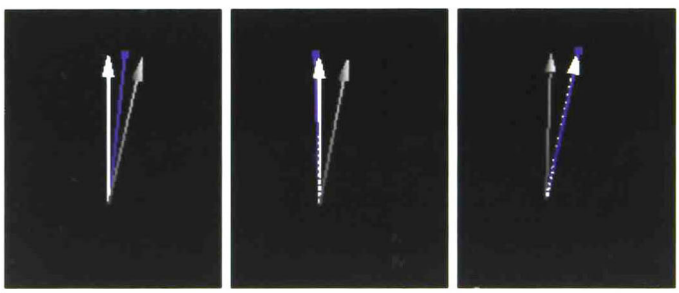
Los movimientos del brazo se expresan mediante transformaciones que son determinadas con motores del tipo expresado en Ecuación 2.50, de igual manera se determinan las transformaciones del objeto observado (que puede ser un tablero de ajedrez) con respecto al sistema de coordenadas de la cámara (Figura 5.8).

Esto permite, conocer el eje del rotor de una manera sencilla mendiante el procedimiento siguiente.

De la ecuación 2.38, tenemos que



(a) Tiempo t (b) Tiempo $t+1$ (c) Tiempo $t+2$



(d) Tiempo $t+3$ (e) Tiempo $t+4$ (f) Tiempo $t+5$

Figura 5.7: Salida de la GRBF-N durante el proceso de aprendizaje. Evolución del posicionamiento: la flecha gris indica el vector de orientación de entrada y la flecha blanca el de referencia. La flecha azul describe la aproximación gradual de la salida de la red neuronal al objetivo.

$$\frac{\mathbf{R} - acos\left(\frac{\theta}{2}\right)}{sen\left(\frac{\theta}{2}\right)} = \underline{l} \tag{5.1}$$

donde R es el rotor R_s. Se usa el álgebra de motores $\mathcal{G}^+_{3,0,1}$.

De esta manera conocemos los pares de líneas en cada movimiento del brazo del robot, en un tiempo t_i, se tiene la línea que representa el eje del rotor trasladado correspondiente a la transformación que permite el movimiento del brazo, la otra línea es aquella que corresponde al rotor trasladado que expresa la transformación del tablero de ajedrez con respecto al sistema de coordenadas de la cámara. Como se puede observar, en esta etapa no es necesario conocer

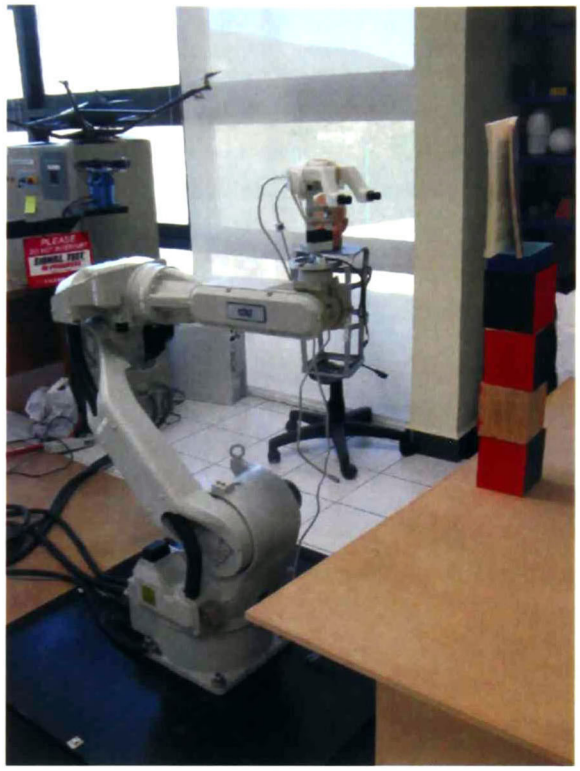


Figura 5.8: Brazo robot con sistema estéreo.

el traslador T_s para obtener el eje del rotor, solamente se conoce el T_c que traslada el rotor en el origen R a una una direccón determinada por t_c. Este rotor R_s expresa una rotación general (Ecuación 2.52). De tal manera que sólo requerimos R_s para calcular su eje.

Es posible determinar un conjunto de líneas a partir de los movimientos (transformaciones) representados por motores. Cada movimiento de la cámara y brazo, es expresado mediante un motor de tal manera que en cada movimiento se genera un par de motores y por lo tanto un par de líneas.

Con estos pares de líneas construimos el conjunto de entrenamiento y lo proporcionamos a la red geométrica para encontrar la transformación faltante.

La red geométrica suministra un motor M como resultado, este motor se obtiene de sus pesos geométricos. Para saber que tan buena es la transfor-

mación calculada, se compara con los métodos clásicos. El motor entonces en términos prácticos se convierte en una matriz homogénea y ésta es usada en matlab para dar un error. El motor encontrado es la transformación X que se desea conocer entre las líneas que son los ejes de los rotores de las transformaciones tanto en el sistema de coordenadas de la cámara como en el sistema de coordenadas del robot. Como se explica en sección 5.2.1, esta ecuación no lineal es conocida como aquella que describe el problema de calibración mano-ojo (hand-eye calibration) en robótica.

En esta conversión entre matrices y motores, se considera la definición $M = TR$ porque de aqui es posible obtener el traslador T_s y el rotor R_s. Recordar que $R_s = T_c R \widetilde{T_c}$, o sea, una rotación general. Además T_s aplica una traslación en dirección del eje del rotor R_s. El R_s es un motor degenerado porque sólo determina una rotación general, se ha trasladado un rotor desde el origen. El procedimiento para calcular el vector de traslación y la matrix de rotación considera lo siguiente.

Sabemos que

$$\begin{aligned} \mathbf{M} &= \mathbf{T}_s\mathbf{R}_s \\ &= (1 + \frac{1}{2}\mathbf{I}\mathbf{t}_s)\mathbf{R}_s \\ &= \mathbf{R}_s + \mathbf{I}\frac{\mathbf{t}_s}{2}\mathbf{R}_s \end{aligned} \tag{5.2}$$

En este momento, se considera la parte dual resultante del motor. Este es el producto geométrico del bivector $\mathbf{t}_s$ y el rotor $\mathbf{R}_s$. Dado que ambos son expresados en términos de la misma base bivectorial, su producto geométrico también será expresado en esta base, el cuál será considerado como un nuevo rotor $\mathbf{R'}_s$. Entonces, se puede escribir

$$\mathbf{M} = \mathbf{R}_s + \mathbf{I}\frac{\mathbf{t}_s}{2}\mathbf{R}_s = \mathbf{R}_s + \mathbf{I}\mathbf{R'}_s \tag{5.3}$$

En este caso, los ejes de línea de los rotores están sesgados, lo cuál significa que ellos representan el caso general de rotores no coplanares. Si la distancia de desplazamiento $\mathbf{t}_s$ es cero, entonces el motor degenerará en un rotor:

$$\mathbf{M} = \mathbf{T}_s\mathbf{R}_s = (1 + \frac{1}{2}\mathbf{I}\mathbf{t}_s)\mathbf{R}_s = (1 + \frac{0}{2}\mathbf{I})\mathbf{R}_s = \mathbf{R}_s \tag{5.4}$$

Cuando dos líneas de ejes del motor son coplanares, se obtiene el motor degenerado. El bivector $\mathbf{t}_s$ puede ser expresado en términos de los rotores usando resultados previos

$$\mathbf{R'}_s\widetilde{\mathbf{R}}_s = \mathbf{M})\frac{\mathbf{t}_s}{2}\mathbf{R}_s\widetilde{\mathbf{R}}_s; \tag{5.5}$$

por lo tanto, el vector de traslación $\mathbf{t}$ puede ser recuperado usando las ecuaciones 5.2 y 5.4, de donde se sigue que

$$\begin{aligned}
\mathbf{R}_s + \frac{1}{2}\mathbf{I}\mathbf{t}_s\mathbf{R}_s &= \mathbf{R}_s + \mathbf{I}\mathbf{R}'_s \\
\frac{1}{2}\mathbf{I}\mathbf{t}_s\mathbf{R}_s &= \mathbf{I}\mathbf{R}'_s \\
\frac{1}{2}\mathbf{I}\mathbf{t}_s\mathbf{R}_s\widetilde{\mathbf{R}}_s &= \mathbf{I}\mathbf{R}'_s\widetilde{\mathbf{R}}_s \\
\mathbf{I}\mathbf{t}_s &= 2\mathbf{I}\mathbf{R}'_s\widetilde{\mathbf{R}}_s \\
\mathbf{t}_s &= 2\mathbf{R}'_s\widetilde{\mathbf{R}}_s
\end{aligned} \tag{5.6}$$

El vector 3D $\mathbf{t}_s$ expresado en la base bivectorial, es referido como el eje de rotación del rotor. Así, $\mathbf{t}_s$ es un bivector a lo largo de la línea del eje del motor. Por lo tanto $\mathbf{t}$, que se le considera como bivector, puede ser calculado en términos de los bivectores $\mathbf{t}_c$ y $\mathbf{t}_s$ como se indica

$$\begin{aligned}
\mathbf{t} &= \mathbf{t}_\perp + \mathbf{t}_\parallel \\
\mathbf{t} &= (\mathbf{t}_c - \mathbf{R}_s\mathbf{t}_c\widetilde{\mathbf{R}}_s) + (\mathbf{t}\cdot\mathbf{n})\mathbf{n} = (\mathbf{t}_c - \mathbf{R}_s\mathbf{t}_c\widetilde{\mathbf{R}}_s) + d\mathbf{n} \\
&= \mathbf{t}_c - \mathbf{R}_s\mathbf{t}_c\widetilde{\mathbf{R}}_s + \mathbf{t}_s \qquad (5.7) \\
&= \mathbf{t}_c - \mathbf{R}_s\mathbf{t}_c\widetilde{\mathbf{R}}_s + 2\mathbf{R'}_s\widetilde{\mathbf{R}}_s \qquad (5.8)
\end{aligned}$$

De esta manera se analiza el motor desde una perspectiva geométrica.

Una vez obtenido el vector de traslación, podemos crear el traslador $\mathbf{T}$ con la ecuación 2.43. Así podemos calcular el motor utilizando las ecuaciones 5.2 y 5.3 quedando de la siguiente manera $\mathbf{M} = (1 + \frac{1}{2}\mathbf{I}\mathbf{t}_s)\mathbf{R}$.

Nótese que estos casos son válidos para un rotor que tenga su eje de rotación en el origen. En el caso de que $\mathbf{R} = \mathbf{R}_{\underline{L}} = \mathbf{T}\mathbf{R}\widetilde{\mathbf{T}}$ el rotor (Ecuación

5.3) devolverá el rotor que tenga el mismo eje pero este estará en el origen, y conservará su ángulo de rotación θ. La ecuación 5.6 devolverá un vector de traslación diferente al del traslador original debido a que se empleó un rotor en el origen en vez de un rotor trasladado fuera del origen. El motor resultante conservará el mismo comportamiento, esto es, sea $\mathbf{M} \in \mathcal{G}^{+}_{3,0,1}$ y de la forma $\mathbf{M} = \mathbf{T_1 R_L} = \mathbf{T_1 T_2 R \widetilde{T}_2}$, entonces podemos encontrar $\mathbf{M_2} = \mathbf{T_3 R_2}$ utilizando las ecuaciones 5.3 y 5.6 donde $\mathbf{M} = \mathbf{M_2}$.

De aqui, se conoce el t_s y el R_s es usado para calcular la matrix de rotación. Tanto el vector obtenido y la matrix se usan para determinar la matrix homogenea de 4x4 elementos.

5.1.1. Discusión

En esta primera etapa hemos diseñado una nueva red usando multivectores para determinar sus parámetros. La principal ventaja de usar rotores ante otras transformaciones valuadas con hiper-complejos es que los rotores pueden ser definidos para cualquier dimensión y un rotor puede rotar *blades* de cualquier grado. Esto es, no únicamente vectores pero también líneas, planos, y cualquier otro objeto geométrico que pueda ser representado por un *blade* puede ser transformado con un *spinor*, es decir, un rotor. En este sentido, las redes valuadas con complejos están limitadas a trabajar con parámetros valuados con complejos mientras que las GRFB-Ns pueden ser usadas para tratar con entidades expresadas con multivectores (hipercomplejos) y esto puede ser hecho únicamente usando AG. Hemos aplicado nuestro esquema a un problema real donde obtenemos los datos de entrada con un sistema de visión estéreo. Se ha mostrado que la GRFB-N puede aproximar la transformación involucrada. Una vez que hemos calculado los rotores (uno para la transformación entre las orientaciones $\mathbf{n}_i$ y otro para la transformación entre las orientaciones $\mathbf{u}_i$) podemos usarlos en otra AG $\mathcal{G}_{p,q,r}$ para rotar diferentes configuraciones de entidades geométricas. El aprendizaje adaptivo de este esquema parece adecuado para ser aplicado a problemas de robótica reales. Es importante explorar más aplicaciones de nuestra GRBF-N tales como el aprendizaje de transformaciones más complicadas como ya hemos mencionado. También, estamos interesados en

aproximar transformaciones entre otras entidades y configuraciones geométricas más complicadas, como la línea en espacio 3-D usando las coordenadas de *Plücker* considerando sus direcciones y momentos. La GRBF-N nos permite estimar la transformación existente entre multivectores en la entrada y salida de la red a través de una combinación de entidades geométricas (rotores) que permite una descripción más natural de la transformación involucrada. La GRFB-N puede calcular la transformación necesitada usando operadores geométricos lo cual representa una ventaja porque esta información es inmediatamente útil, no requiere algún procesamiento posterior. No existe alguna red neuronal que permita suministrar la transformación geométrica mediante operadores geométricos como nuestro modelo.

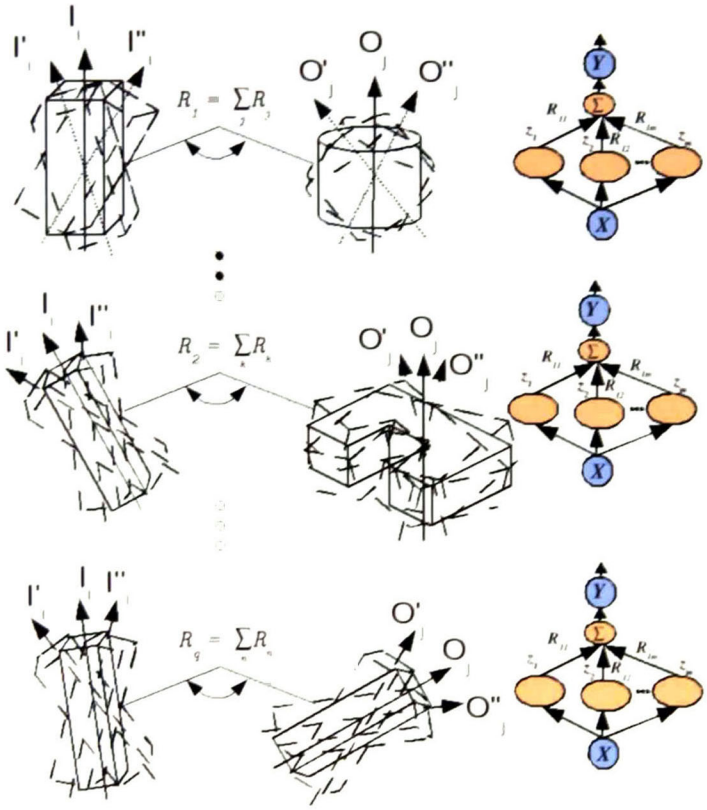


Figura 5.9: Diferentes tipos de transformaciones estimadas por la GRBF-N

5.2. La GRBF-N construida con motores

Hemos actualizado nuestro modelo que trabaja con rotores. Esta vez representamos los pesos de la red usando motores para considerar las transformaciones de translación y rotación simultaneamente y de manera compacta.

5.2.1. Calibración *Hand-Eye*

Nuestra red nuevamente es aplicada a tareas relacionadas con el área de robótica guiada visualmente tal como resolver el problema de Calibración Mano-Ojo (*Hand-Eye*) que involucra a una cámara adjunta a un brazo de robot que será dirigido hacia una meta (Figura 5.11). Las cámaras capturan las señales visuales en el espacio visual 3D y emplean su propio sistema de coordenadas de referencia del mundo para representar estas señales. Entonces, el problema de calibración "mano-ojo" surge cuando se intenta determinar la transformación de grupo entre las coordenadas de referencia del dispositivo mecánico y el marco de coordenadas de la cámara. La metodología clásica para describir el problema de calibración mano-ojo en términos matemáticos es usando matrices homogéneas de transformación. La ecuación matricial de una transformación Euclidiana es

$$AX = XB \tag{5.9}$$

donde las matrices $A = A_1 A_2^{-1}$ y $B = B_1 B_2^{-1}$ expresan la eliminación de la transformación entre los dos sistemas de coordenadas referidos. De la ecuación (5.9), la siguiente ecuación de matrices y una de vectores puede ser derivada dividiendo la transformación Euclidiana en los componentes de rotación y translación:

$$R_A R_X = R_X R_B, \tag{5.10}$$

$$(R_A - I)t_X = R_X t_B - t_A. \tag{5.11}$$

En la literatura encontramos una variedad de métodos para estimar R_X de la ecuación (5.10), la mayoría de ellos estiman primero la matriz de rotación separada del componente de translación. Para calcular las transformaciones de rotación y translación de manera compacta, existe una propuesta que usa

álgebra de motores. Ésta considera usar movimientos de líneas en términos de motores [48].

5.2.2. Resolviendo AX=XB usando el álgebra de motores

En términos de motores, la ecuación del sistema (5.9) puede ser expresada como

$$M_A M_X = M_X M_B \tag{5.12}$$

o como

$$M_A = M_X M_B \widetilde{M}_X \tag{5.13}$$

donde $M_A = A + IA'$, $M_B = B + IB'$, y $M_X = R + IR'$ Esta ecuación puede ser simplificada para mostrar la relación de motores entre el eje línea del motor de la cámara L_{Ai} y el eje línea del motor de la mano L_{Bi}. De acuerdo al teorema de Chen [49], la ecuación del problema de calibración mano-ojo se reduce a

$$L_A = M_X L_B \widetilde{M}_X \tag{5.14}$$

que muestra que en este tipo de formulación de problema, la rotación y *pitch* de M_A y M_B son siempre iguales a través de todos los movimientos de la mano y por lo tanto pueden ser ignorados en los cálculos. Bastará considerar sólo los ejes de rotación de los motores involucrados; esto es, la ecuación (5.14) es reducida a únicamente el movimiento de los ejes línea de la mano L_B hacia los ejes línea de la cámara L_A. Gracias al uso del álgebra de motores, es posible la simplificación del problema de calibración mano-ojo. Este problema se reduce a sólo considerar el movimiento de las líneas de ejes. Una vez que la ecuación de este problema ha sido reducida, ella depende únicamente de bivectores 3D.

5.2.3. Resolviendo AX=XB usando álgebra de motores y la GRBF-N

Para la estimación del movimiento rígido desconocido, empleamos la GRFB-N, que es realmente una propuesta con multivectores que definen a los motores. Se tienen n movimientos 3D descritos por líneas, y cada una tiene

seis parámetros. Nuevamente, los datos son adquiridos usando un sistema de visión estereo (Figura 5.11). Se tiene un objeto (tablero de ajedrez que no se mueve). Primero, el objeto debe ser detectado en ambas cámaras y capturarlo con cada una. Usamos un algoritmo basado en puntos característicos para la detección y registro de objetos planares usando imágenes [50]. Con técnicas de visión, específicamente de calibración de la cámara, conocemos las transformaciones entre el sistema de coordenadas del objeto y el de la cámara izquierda. Estas transformaciones posteriormente son expresadas a través de motores y se usan líneas para representar cada movimiento determinado por un motor cada cierto tiempo. Esto es posible dado que el brazo robótico se está moviendo y el sistema estéreo está montado sobre su efector final. Las líneas relacionadas con los movimientos del efector final del brazo también son expresadas con motores y son determinados en cada tiempo t_i. Para obtener las matrices de transformación de cada movimiento usamos la representación de transformaciones de Euler ZYZ dado que el software del robot que usamos ejecuta el movimiento del brazo mediante una sexteta de parámetros (Figura 5.10), posteriormente son convertidas a motores. Entonces, tenemos un conjunto objetivo de líneas (l_{Bi}) y otro conjunto de inicial l_{Ci} como se observa en la figura 5.11(a). Supongamos que L_{Ci} son las líneas que expresan el movimiento de la cámara izquierda y L_{Bj} las líneas que representan los movimientos del efector final, las líneas son determinadas usando multivectores de seis dimensiones. Entonces, construimos pares de entrada (para el entrenamiento y prueba) usando las líneas obtenidas por los movimientos del efector final. Hemos determinado L_{Ci} para X_i y L_{Bj} para Y_i como entradas y objetivos de la GRBF-N, respectivamente. Usamos estos conjuntos de datos para entrenar y probar la GRBF-N. La figura 5.12 muestra que los valores del motor convergen a un valor, lo cual es observado en esta figura porque las salidas de la GRBF-N están muy cerca de las líneas objetivo. En estos experimentos estamos usando multivectores de seis dimensiones para determinar las transformaciones (que expresan el movimiento) del efector final y cámara. Consideramos que el error (distancia Euclidiana entre las líneas objetivo y las líneas de salida de la GRBF-N) debe ser menor a 0.0001, y usamos los valores $\alpha = 0.15155, \beta = 0.07123108, \eta = 0.9$, cinco centroides fueron requeridos esta vez, una entrada y una salida.

El algoritmo se resume como sigue:

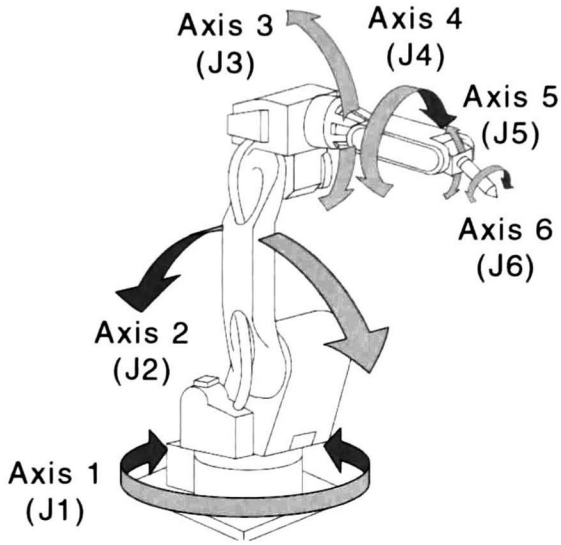
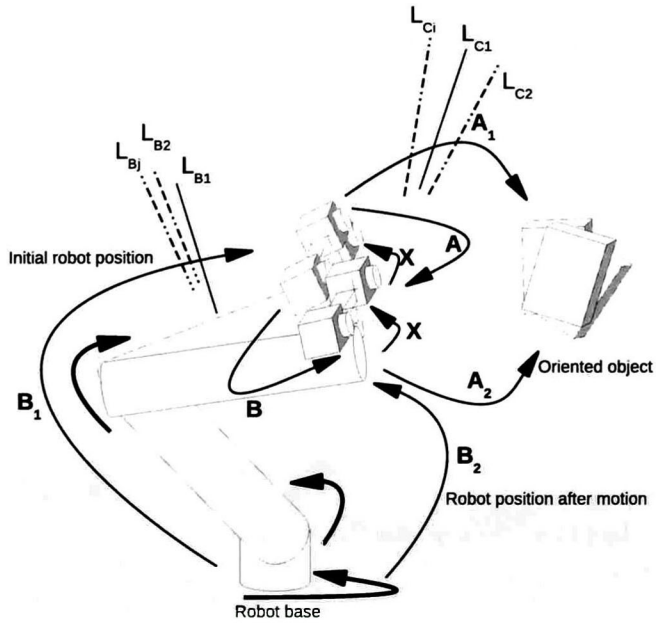


Figura 5.10: Brazo industrial AdeptSix 600.

1. Considera n movimientos de la mano (B_i) y sus respectivos movimientos de la cámara (C_i). Entonces, se extraen las direcciones y momentos de línea para construir el conjunto de entrada y objetivo para la red geométrica.

2. Aplicar la GRBF-N usando estos conjuntos para obtener el motor M_X que representa la transformación **X** (Ecuación 5.9).

Note que los métodos clásicos no pueden ser aplicados en tiempo real. Necesitan acumular un conjunto largo de entradas para aplicar posteriormente un procedimiento SVD. En el caso de [1], las líneas son usadas para construir una matriz que es usada para obtener la transformación final. Es importante aclarar que en estos experimentos cada entrada de la red geométrica depende de objeto detectado (tablero de ajedrez). Dado que usamos un sistema estereo, esta detección es afectada por condiciones de iluminación por lo que movemos el brazo del robot a través de una trayectoria por tiempos (t_i). En este sentido, L_{Ci} y L_{Bi} son calculados cada tiempo t_i en el que una nueva posición del efector final es determinada. El tiempo de detección del objeto es de alrededor de 30 milisegundos, esto significa que cuando el brazo robótico está en una



(a)

(b)

Figura 5.11: Sistema estéreo montado en el efector final del brazo robótico. (a) Un esquema representativo. (b) Área de trabajo real.

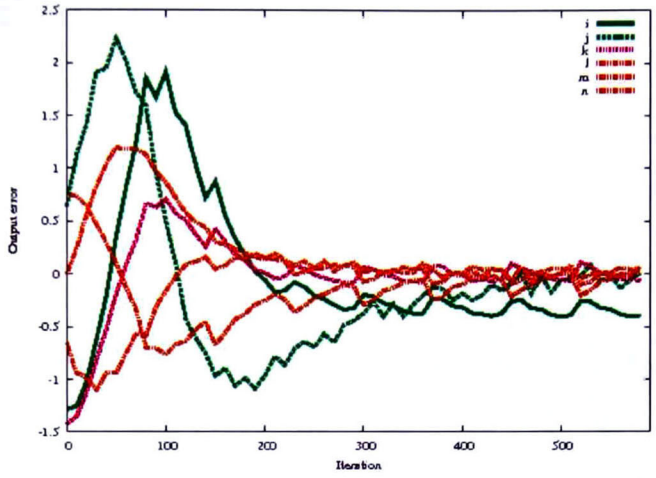


Figura 5.12: Error de salida durante el proceso de aprendizaje.

nueva posición específica, el sistema espera y detecta un objeto o se mueve a otra posición. Aqui, el cálculo de las líneas relacionadas con el movimiento de la cámara izquierda y del brazo es más rápido que el movimiento del brazo. Considerando un tiempo promedio para calcular cada par de entrada (línea objetivo y origen), el sistema requiere cerca de 32 milisegundos cada vez que una nueva posición es alcanzada por el brazo. Note que también existe un tiempo variante específico requerido por el sistema para mover el brazo de una posición en el tiempo t_i a una nueva posición en el tiempo t_{i+1}, esto es, cada posición es alcanzada requiriendo tiempos diferentes donde el brazo se mueve a una velocidad constante de 20 mm/segundo que permite el brazo AdeptSix 600. En el proceso completo, el sistema requiere cerca de 10 minutos para calcular una aproximación de la transformacióon requerida para lograr un pequeño error. El mínimo número de movimientos requeridos para obtener un buen resultado en esta tarea depende el tipo de movimientos. Se recomienda parar el entrenamiento de la GRBF-N cuando el error es pequeño y se tenga una transformación satisfactoria. En estos experimentos generamos 100 movimientos y mientras se genera un movimiento se le agrega ruido Gaussiano (en-línea) de manera que al final se tiene un conjunto de 500 pares de entrenamiento. Estos 500 pares fueron alimentados a la GRBF-N y

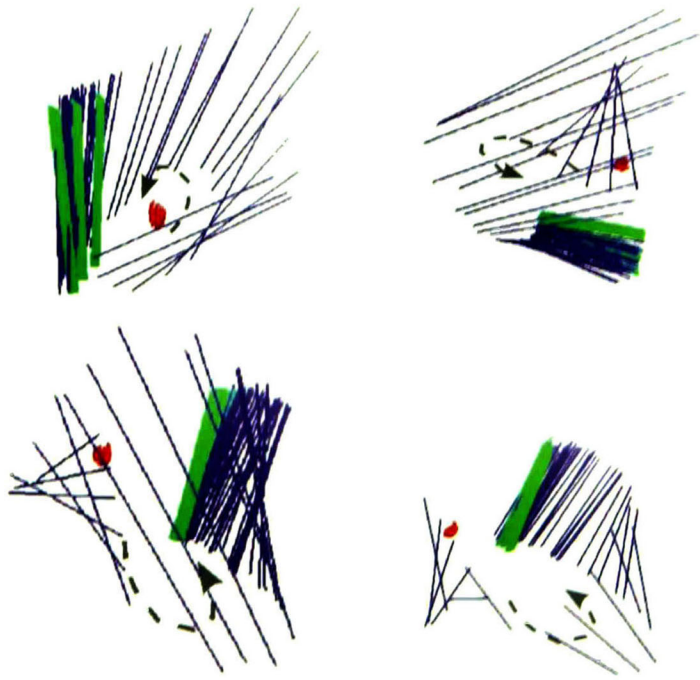


Figura 5.13: Diferentes vistas de la salida de la GRBF-N durante el proceso de aprendizaje. La salida de la GRBF-N (líneas azules). La red codifica la transformación geométrica existente entre el conjunto de líneas rojas y las verdes.

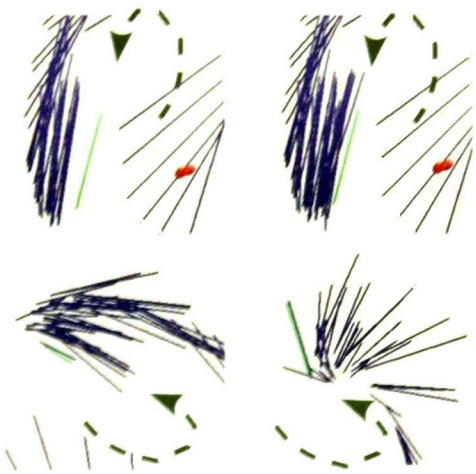


Figura 5.14: Ampliación de la zona objetivo de la figura 5.13. Salida de la GRBF-N (línea azul), y la línea objetivo (verde). Mientras la salida de la GRBF-N es diferente al objetivo, sus parámetros son ajustados hasta tener una orientación de línea similar a la deseada. Finalmente, se encuentra la aproximación de la transformación geométrica entre las líneas origen y objetivo (roja y verde, respectivamente).

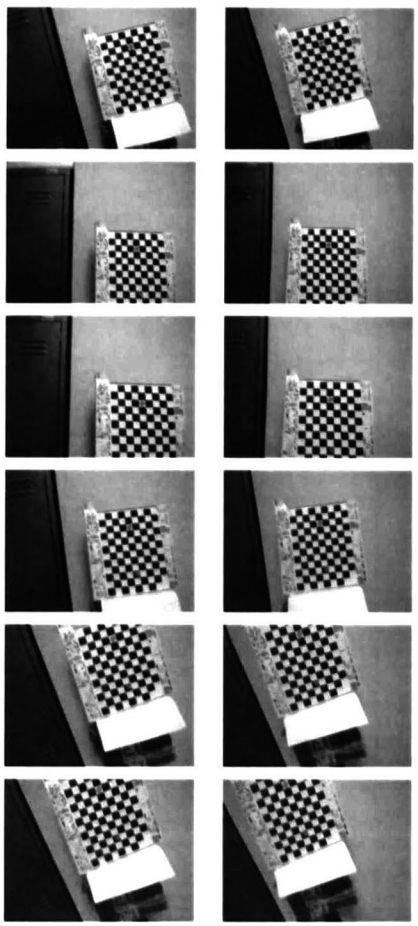


Figura 5.15: Diversas tomas del tablero de ajedrez conforme el efector final del robot se mueve.

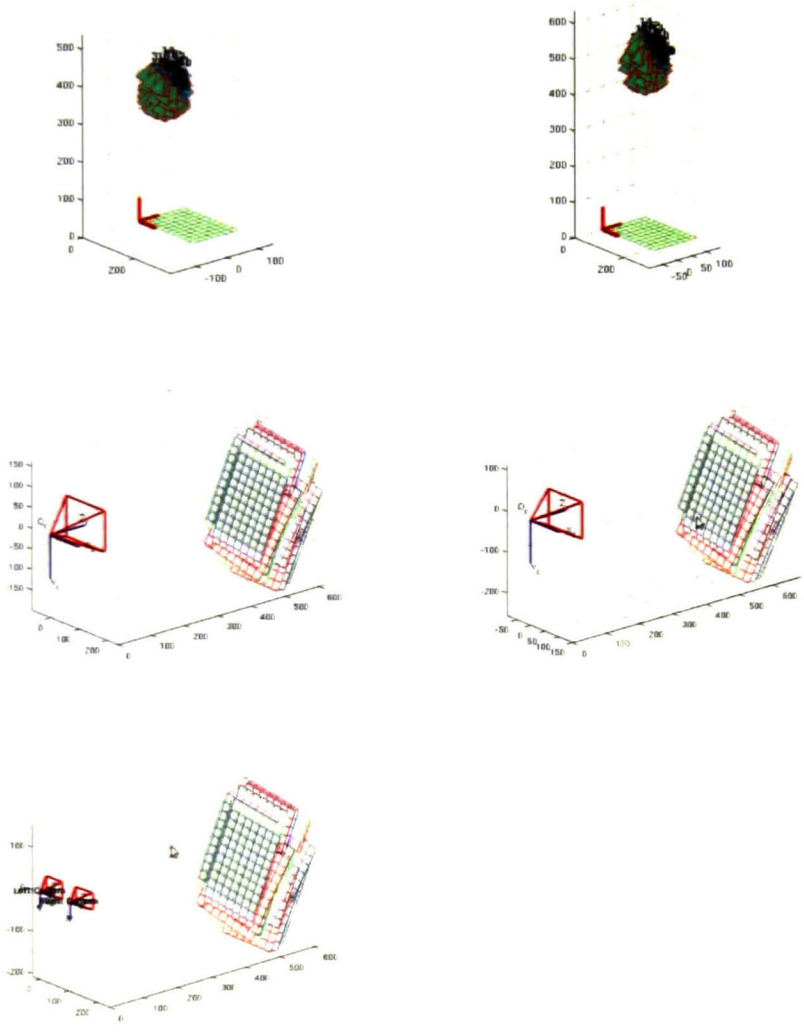


Figura 5.16: Movimientos del sistema estéreo a partir del movimiento del efector final del brazo.

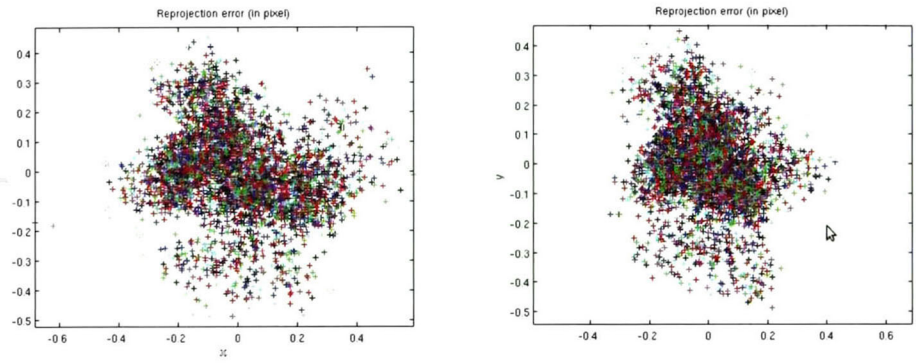


Figura 5.17: Errores de calibración en cámaras.

después de cerca de 250 iteraciones (Figura 5.12) obtenemos una aproximación satisfactoria de la transformación que resulta en un error pequeño. En la figura 5.15 se muestran algunas imágenes del tablero de ajedrez tomadas durante el movimiento del brazo robótico. La figura 5.16 muestra estos movimientos gráficamente. Se observan los movimientos desde la perspectiva del mundo y desde el marco de referencia de cada cámara (izquierda y derecha). Finalmente se observan los movimientos desde la perspectiva del sistema estéreo. La figura 5.17 indica los errores de calibración en cada cámara. El proceso de aprendizaje nos permite observar que al inicio del mismo la salida de la GRBF-N es completamente diferente de la salida esperada. Las figuras 5.13 y 5.14 muestran este proceso. La figura 5.13 muestra la evolución geométrica de la línea en la salida de la GRBF-N durante el proceso de entrenamiento. Las pequeñas líneas (rojas) indican la orientación inicial y las grandes (verdes), indican la orientación de línea objetivo, ambos conjuntos concentrados en áreas pequeñas. Puede ser observada la manera en la que la salida de la GRBF-N (línea azul) está cambiando durante el proceso de aprendizaje. Cada una de estas líneas indica el cambio de los motores (pesos) de la GRBF-N que es ncesario para lograr la orientación de línea deseada. Ambas figuras muestran la evolución de la línea pero en perspectivas diferentes. La figura 5.14 muestra una ampliación de diferentes perspectivas de esta evolución. Las figuras 5.12 y 5.13 muestran que nuestra red estima adecuadamsnte la transformación re-

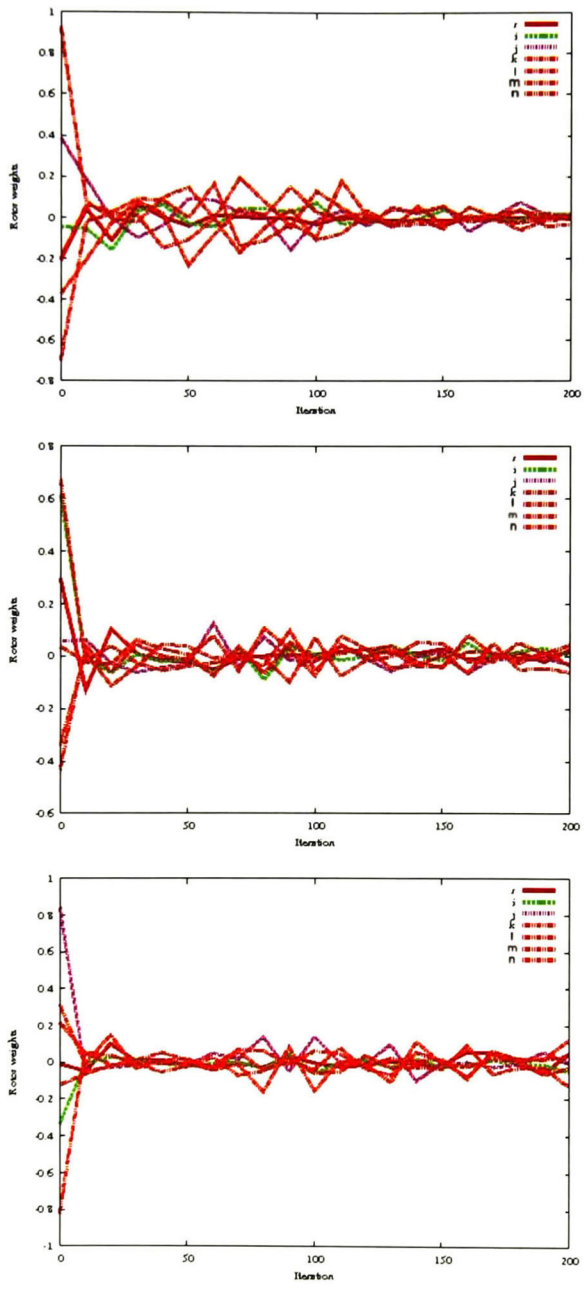


Figura 5.18: Convergencia de red neuronal geométrica para el problema de Calibración Mano-Ojo

querida para lograr la orientación objetivo a partir de una orientación inicial. Conforme la red aprende, su salida es más cercana a la meta deseada. También, nuestro esquema converge de manera que usando diferentes instancias de datos de prueba, obtenemos un error de 0.3, lo cual representa la distancia Euclidiana entre la línea de salida de la GRBF-N y la línea objetivo. La figura 5.18 muestra el comportamiento de la red neuronal para tres ejecuciones con mismos datos del problema. El M_X encontrado por la red GRBF-N es el *motor* requerido para transformar una entidad en el sistema de coordenadas de la cámara al sistema de coordenadas del efector final. Definamos M_w que sea el *motor* que se requiere para transformar el sistema de coordendas del efector final en el sistema del mundo, entonces se requiere el *motor* completo:

$$M_f = M_w M_X \tag{5.15}$$

para realizar una transformación desde el sistema de coordenadas de la cámara al del mundo. Entonces, para conocer que tan útil es la transformación encontrada por la GRBF-N, se calculó una línea en el espacio 3D (con ayuda del tablero de ajedrez) usando el sistema estéreo; y por otro lado, se tiene la verdadera línea en el mundo. Aplicamos la transformación usando M_X sobre la línea determinada por el sistema estéreo, y posteriormente aplicamos la transformación M_w. Ahora tenemos la línea cerca de la línea que se localiza en el sistema de coordenadas del mundo. Esto indica que nuestro esquema trabaja bien con un error determinado mediante una distancia Euclidiana, es de 25 mm (el punto más cercano entre ambas líneas atendiendo que su orientación es similar). Mostramos nuestro espacio de trabajo en la figura 5.11(b). Se observa que la estructura algebraica de las entidades y operadores nos ayuda a formular nuestra propuesta para estimar la transformación requerida. Y nuevamente, el AG permite determinar entidades geométricas como líneas, y motores de manera que podemos tener transformaciones entre las líneas usando multivectores. Esta propuesta puede ser usada para calibrar una variedad de sistemas donde se presente un problema similar a la calibración hand-eye, tales como los subsistemas de un humanoide, donde hay muchas partes cuyos ejes puede ser expresados por líneas. En este sentido, esta red puede encontrar la transformación necesaria para mover partes de brazos robóticos o piernas guiados por visión. El error final (entre las orientaciones reales y calculadas) es

suficiente para este tipo de aplicaciones donde se requiere conocer únicamente la zona objetivo para guiar al brazo robótico. Nuestra método puede ser útil para tareas donde un robot (brazo, pierna, mano, dedo) debe moverse, pero sin gran precisión. La red GRBF-N es capaz de estimar transformaciones rígidas.

5.2.4. Discusión

Nuevamente, en estos últimos experimentos usamos el AG para cálculos en robótica guiada con visión. Hemos usado el álgebra de motores pues el problema que tratamos implica el álgebra de líneas. Aplicamos nuestra GRBF-N a un problema que implica obtener datos de entrada con un sistema estéreo y cinemática. La GRBF-N nos permite aproximar la transformación existente entre las líneas en su entrada y salida, a través de una combinación de motores que permiten una descripción más natural de la estimación requerida. Esta transformación expresada por motores representa una ventaja porque es inmediatamente útil ya que no requiere de procesamiento posterior. Estos motores son actualizados cada vez que la GRBF-N tiene una línea de entrada de manera que si existe una transformación, los motores serán ajustados hasta que la verdadera transformación se encuentre. El aprendizaje adaptivo de este esquema es útil en aplicaciones donde los parámetros de tipo motor son estimados en tiempo real. Es importante indicar que todas las transformaciones en este problema han sido estimadas usando álgebra de motores. Nuestro enfoque es más eficiente que los métodos *batch* estándar porque trabaja en tiempo real, estimando la transformación rígida bajo perturbaciones temporales. En contraste, los métodos estándares requieren calibrar en cada momento, primero colectando datos, y luego calculando un procedimiento *batch*, frecuentemente usando técnicas de optimización o SVD. Estamos considerando mejorar la precisión de nuestro método para aplicarlo a tareas donde se requiera alta precisión. Si los conceptos geométricos en los que se basa el diseño son bien fundamentados y correctos, siempre es posible en un futuro la mejora del desempeño de la red con nuevos algoritmos de entrenamiento.

Existen muchos aspectos que nuestro método ofrece que ninguno de los ya existentes considera. Nuestra propuesta permite realizar el cálculo en un

mismo lenguaje geométrico. Esto facilita la comunicación entre el módulo de entrada, de proceso y de salida de cualquier sistema. En este caso, las entidades observadas en el medio ambiente (líneas, puntos, círculos, etc.) son definidas en lenguaje geométrico y nuestro algoritmo manipula tales entidades usando operadores geométricos en el mismo lenguaje matemático de manera que la salida del algoritmo también está definida en ese mismo lenguaje.

La característica fundamental de este tipo de arquitectura es que las operaciones geométricas son simples en este lenguaje y es sencillo implementar ideas que tienen una naturaleza geométrica. Otro aspecto importante es que la arquitectura de la red permite extraer la transformación geométrica entre la entrada y la salida, de manera que cualquier entrada posterior al entrenamiento será transformada usando los motores ajustados en la arquitectura y esto no es posible usando redes neuronales convencionales.

5.3. La RGRBF-N usando motores

Se ha trabajado con diversas variantes para el entrenamiento de redes recurrentes para reconocer secuencias temporales de líneas (Figuras 4.6, 4.7, 4.9, 4.10). La autoconexión de cada neurona de entrada, la conexión de las neuronas ocultas y de salida a las de contexto permite cierta memoria a las neuronas. La autoconexión permite a estos modelos considerar las entradas previas y no únicamente las entradas en el instante t. Cada entrada $\mathbf{L}_i$ representa la ocurrencia de una línea i de una secuencia. Los coeficientes de cada uno de los 6 elementos de la definición de una línea sólo pueden ser 1 o 0 en estos experimentos. No se consideran coeficientes diferentes de 1 ó 0. La ecuación 4.7 define la línea mediante bivectores para calcular la dirección de la línea y el momento.

5.3.1. La RGRBF-N tipo Elman

Este modelo se probó para el reconocimiento de secuencias temporales de líneas. Las secuencias temporales analizadas son compuestas de 5 líneas.

Durante el proceso de entrenamiento, las líneas son presentadas a la red, una a una y se determina un elemento geométrico al que se asocian (que podría ser la última línea de la secuencia).

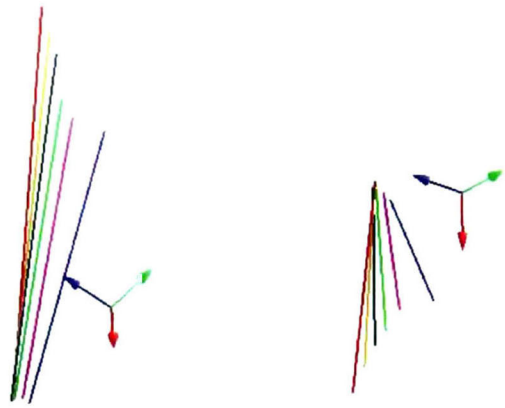


Figura 5.19: Secuencia de líneas con transformación variante en el tiempo. Secuencia de líneas codificada.

En la figura 5.19 se puede apreciar una secuencia de líneas. Cada secuencia inicia y termina con una misma línea terminal; se usa una línea vertical en nuestro caso. Para cualquier otra línea que no sea la inicial o final, el orden de su orientación dentro de la secuencia debe ser coherente con el movimiento suave de la línea. El trabajo a futuro es actualizar el modelo para considerar cualquier tipo de orientación en la secuencia. Para esta actualización, se consideraría también el fundamento teórico de que con suficientes unidades en la red, la longitud de la cadena no tiene límite. La arquitectura implica una única neurona de entrada, 20 neuronas ocultas y 25 neuronas de contexto y 1 de salida (Figura 4.6(c)). El número de neuronas en la capa oculta y de contexto, depende de la cantidad de líneas en la secuencia aprendida por la red.

5.3.2. La RGRBF-N basada en la autoconexión

Este modelo permite presentar a la red una secuencia completa en cada instante de tiempo, es decir, un conjunto de líneas de manera que una secuencia

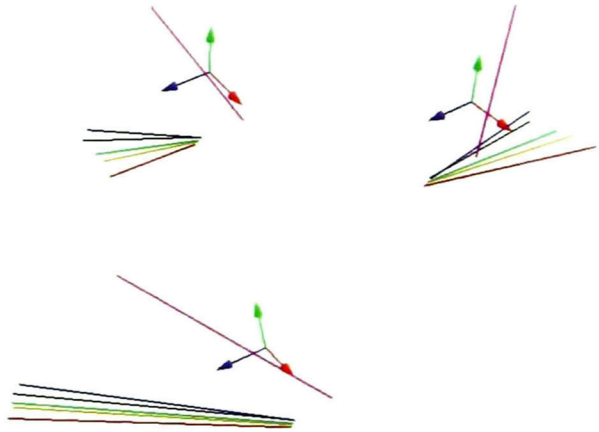


Figura 5.20: Secuencia de líneas con transformación variante en el tiempo. Secuencia asociada a un elemento geométrico (línea).

se asocia a una entidad geométrica determinada en la neurona de salida (Figura 4.10). La figura 5.20 muestra las asociaciones de una secuencia de líneas con otra línea. Tal asociación (que es una transformación) es capturada por la red geométrica. El error de entrenamiento de la red se puede visualizar en la figura 5.21.

La arquitectura implica 5 neuronas de entrada, dado que 5 líneas definen la secuencia; 20 neuronas ocultas, 25 neuronas de contexto y 1 de salida (Figura 4.10(b)). El número de neuronas en la capa oculta y de contexto, depende de la cantidad de líneas que determinan una secuencia aprendida por la red asi como tambiń de la cantidad de secuencias. Cada neurona de la capa oculta memoriza un prototipo (vector de la secuencia), y cada neurona de la capa de salida representa una clase o categoría (secuencia).

Los parámetros que se requiere definir son pesos de autoconexiones de las neuronas de entrada (Ecuación 5.16) y la máxima distancia de campos de influencia de neuronas de la capa oculta. La inicialización de los motores se determina usando números de valores iguales, al finalizar el entrenamiento estos valores del motor son diferentes. Dos secuencias han sido aprendidas por la red. Después de la etapa de entrenamiento, todas las secuencias aprendidas

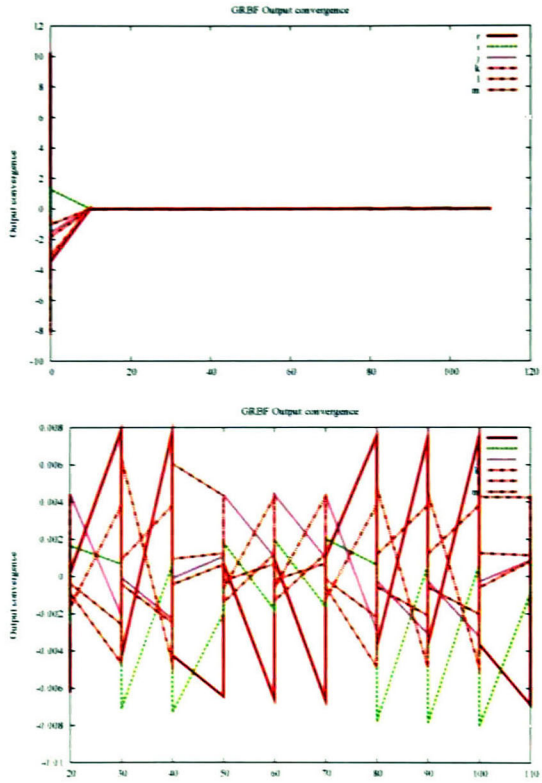


Figura 5.21: Evolución de salida del error de la RGRBF-N (Arriba). Acercamiento (abajo).

son reconocidas con éxito, y el fenómeno de sobre-entrenamiento encontrado en el algoritmo de *back propagation* no tiene efecto en esta red.

$$\mathbf{M}_i = (0.8 + 0.8 * e_2e_3 + 0.8 * e_3e_2 + 0.8 * e_2e_1) + I(0.8 + 0.8 * e_2e_3 + 0.8 * e_3e_2 + 0.8 * e_2e_1) \quad (5.16)$$

Un experimento podría ser aplicar la red geométrica para estimar el movimiento relativo entre la articulación final de un brazo de robot y una línea en 3D perteneciente a un objeto rígido. Los parámetros de la línea en 3D se recuperan durante el movimiento del brazo usando un sistema de visión estéreo. Este enfoque puede ser usado para varios tipos de aplicaciones industriales. El sistema podría observar un par de líneas sobre el piso y se movería en espacio 3D, siempre conservando aquellas líneas en su campo de visión. La principal tarea sería estimar automáticamente el movimiento relativo entre las líneas del piso y la articulación final del sistema. El sistema de visión puede consistir de dos cámaras CCD de 640x480 pegadas a la última articulación del brazo de robot. El brazo de robot tiene seis articulaciones que pueden ser controladas por seis variables (*x,y,z,roll, pitch* y *yaw*) como en nuestros previos experimentos. Las coordenadas (*x,y,z*) describen la posición del sistema de coordenadas de la herramienta, *T*, de la articulación final, la cuál se refiere al sistema de coordenadas global, *W*, del robot. La orientación de la articulación final es descrita por las variables (*roll, pitch, yaw*) en términos de ángulos de Euler. El movimiento del brazo de robot es controlado por la posición y orientación entre el sistema de coordenadas de la herramienta *T* y el sistema base *W*. El procedimiento de calibración de la cámara obtiene la matriz de transformación proyectiva **P**, la cuál relacionada el espacio visual al plano de imagen up to scalar value. El sistema de coordenadas de la cámara *C* en la articulación final es relacionado al marco del sistema de herramienta *T* vía una transformación **X**, la cuál fue calculada usando el procedimiento de calibración hand-eye. Cuando el sistema *T* es transformado de T_1 a T_2 mediante **T**, el sistema de la cámara **C** será transformada de $\mathbf{C}_1$ a $\mathbf{C}_2$ por una transformación **C** como lo indica la ecuación 5.17.

$$\mathbf{C} = \mathbf{XTX}^{-1} \quad (5.17)$$

Dado que el movimiento del brazo de robot especificado por la trans-

Cuadro 5.1: Procedimiento para estimación de movimiento

Algoritmo EstimaMovimiento()

1. Calibración de cámara para obtener **P** para cada cámara
2. Calibración Hand-eye para obtener **X** para cada cámara
3. Se toman imágenes con el movimiento del brazo del robot con una frecuencia de muestreo constante
4. Extracción de líneas a partir de los motores
5. Se estima la transformación usando las observaciones de líneas representadas con álgebra geométrica y la red neuronal geométrica

formación **T** es conocido, podemos compararlo con el movimiento relativo de **C**. El movimiento relativo del marco W y el marco C es un movimiento tornillo con velocidad angular constante $w = -\phi/90$ y velocidad de traslación constante $\mathbf{v}_s = 0.2$. La línea eje, $\mathbf{L}_s$, es paralela al eje z del sistema C, y un punto tocando esta línea eje está dado por ciertas coordenadas. En álgebra de motores, $\mathcal{G}^+_{3,0,1}$, la línea eje está dada por $\mathbf{L}_s$ (ver ecuación 5.18).

$$\mathbf{L}_s = e_1e_2 + \mathbf{I}(1{,}5e_2e_3) \wedge (e1_e2) = e_1e_2 + \mathbf{I}1{,}5e_3e_1 \tag{5.18}$$

y el motor **V** es calculado como

$$\mathbf{V} = (1 + \mathbf{I}\mathbf{v}_{s0}/2)(cos(\omega/2) + sin(\omega/2)\mathbf{L}_s) \tag{5.19}$$

$$= 0.9994 - 0.0349e_1e_2 + \mathbf{I}(0.0035 - 0.0523e_3e_1 + 0.0999e_1e_2) \tag{5.20}$$

El motor $\mathbf{M}_{i+1}$, expresado en términos de álgebra lineal, está dado por

$$\mathbf{M}_i = \mathbf{V}_{\mathrm{MI}}\mathbf{M}_{i-1} \tag{5.21}$$

inicializado con $\mathbf{M}_0 = (1, 0, 0, 0, 0, 0, 0, 0)^T$. Las líneas en 3D reconstruidas serían usadas para estimar el movimiento relativo entrena la articulación final y el objeto sobre el piso usando la red neuronal.

El procedimiento planeado para este experimento se resume en el Cuadro 5.1. El algoritmo para la estimación de movimiento se ejecutaría en línea recursivamente siguiendo los pasos del 3 a 6.

El objetivo es que los ocho parámetros estimados a partir del movimiento, a lo largo del entrenamiento se acerquen a un valor específico de tal manera que la red neuronal aproxime la transformación deseada que es expresada mediante los 8 parámetros del motor.

Este experimento es un reto para nuestro último algoritmo presentado de la red neuronal geométrica, el cuál es una herramienta apropiada para la estimación de las transformaciones de tornillo usando observaciones de líneas.

5.3.3. Discusión

Nuestro diseño de la red recurrente geométrica definida con motores permite reconocer secuencias temporales de líneas. La mayor ventaja de este tipo de red es la flexibilidad del proceso de entrenamiento: tiempo de entrenamiento relativamente corto y pocos parámetros para optimizar. La red RGRBF-N es inspirada por las ventajas de las redes RBF y las redes recurrentes. Nuestro trabajo a futuro será centrado en el desarrollo matemático relativo a la aplicación de la RGRBF-N en problemas complejos como la vigilancia de sistemas industriales (detección de fallas, localización y diagnóstico) donde el tiempo juega un papel muy importante en la evolución del sistema. Existen muchas variantes de arquitecturas y reglas de aprendizaje de las redes recurrentes. Éstas comparten la propiedad de ser capaces de usar y crear internamente estados reflejando dependencias temporales o estructurales. Para simples tareas, la organización del espacio de estados sencillamente refleja los componentes de los datos de entrenamiento. El espacio de estados es en este caso valuado con multivectores. Esto significa que las sutilezas más allá de los componentes, como variaciones estadísticas, pueden influenciar la organización del espacio de estados. Para tareas más difíciles (donde se requiera quizás un rastreo más amplio de memoria y la dependencia del contexto es evidente), el espacio continuo y altamente nolineal, ofrece nuevos tipos de dinámica. El análisis de las representaciones internas aprendidas y procesos/dinámicas es crucial para nuestro entendimiento de lo que y cómo éstas redes procesan. Es útil mencionar que muchos problemas del mundo real que uno podría pensar que requieren arquitecturas recurrentes para su solución son solucionados con ar-

quitecturas de varias capas por lo que es recomendable primero analizar si el problema no puede resolverse sin recurrencia.

En este trabajo se modeló el movimiento de líneas en el espacio 4D usando álgebra de motores. Este tipo de modelado linealiza la transformación Euclidiana de movimiento rígido 3D. El modelo de movimiento de líneas usando motores es muy atractivo para diseñar una red neuronal geométrica para estimar el movimiento. Para el diseño de los modelos recurrentes, comenzamos con una arquitectura estática y usando rotores, posteriormente como una extensión natural del proceso, se usaron motores en los pesos y se obtiene una arquitectura recurrente. Nuestros modelos de red con motores, tienen la virtud de que pueden estimar simultáneamente las transformaciones de rotación y translación. Dado que la mayoría de los algoritmos recursivos en la literatura calculan las transformaciones de rotación y translación separadamente, podemos afirmar que ésta es una de las ventajas más importantes de nuestros modelos de redes neuronales. El modelo dinámico de movimiento usando motores como estados es útil para efectivamente formular y calcular el movimiento de tornillo de una línea como una entidad rgída mínima. Usando redes neuronales convencionales no se puede leer las transformaciones a partir de los pesos de la red.

En los algoritmos de las redes neuronales geométricas, modificamos el paso de estimación tal que ciertas restricciones geométricas son satisfechas, lo cuál hace la convergencia de la estimación más rápida al estado de motor apropiado. Las pruebas con datos simulados confimaron que el ajuste de los pesos de la red mejoraron y son únicos frente a las capacidades de una red neuronal clásica.

Presentamos una aplicación real de manipulación de robot guiada con visión. El sistema fue eficientemente calibrado usando movimientos de robot controlados y nuestro efectivo método de calibración hand-eye. La recuperación de los parámetros de las líneas en 3D fue llevada a cabo usando un sistema de visión estéreo y otras técnicas de visión. Durante el movimiento del robot, la red neuronal eficientemente estimó el movimiento relativo entre su articulación final y un objeto 3D. Estos experimentos confirmaron que nuestro algoritmo es un método atractivo en línea para estimar movimientos tornillo

usando observaciones de líneas.

Si modelamos las redes usando el álgebra geométrica conformal $\mathcal{G}_{4,1}$, podremos extender la transformación de rígida a conformal. Así se podrá leer de la red geométrica motores afinos, los cuales son muy frecuentes en procesos reales donde los movimientos de las líneas evolucionan siguiendo otro tipo de transformaciones.

Capítulo 6

CONCLUSIONES

La manera de pensar y modelar en el marco de trabajo del AG es bastante natural. Con el AG, las operaciones algebraicas tienen una interpretación geométrica. Las ecuaciones compactas son fáciles de interpretar y las escenas complicadas pueden ser tratadas. El álgebra indica al desarrollador los conceptos a usar pues permite visualizar un paradigma geométrico. Esta investigación desarrolla algoritmos de redes neuronales cuyo proceso de aprendizaje se realiza en el contexto del AG. Las entradas, los pesos, la actualización de los pesos durante la etapa de aprendizaje, y las salidas de la red son representadas usando el lenguaje del AG.

El desarrollo de algoritmos de esta naturaleza es un avance importante en la solución de problemas donde los números hipercomplejos describen de mejor forma ciertos fenómenos físicos. De manera tal que es importante que todo el proceso de aprendizaje se desarrolle en el mismo marco por cuestiones de intuición, claridad y simplicidad. Se aprovechó que las expresiones en AG normalmente tienen baja complejidad simbólica.

Las redes geométricas diseñadas son capaces de realizar un entrenamiento y encontrar una transformación descrita con operadores geométricos que puede ser usada en el contexto puramente geométrico ya que éstos presentan una facilidad de representación en el espacio tridimensional. Tratamos datos expresados en alto nivel, como son líneas, círculos, planos. Para los diseños analizados hemos usado motores, rotores y transladores.

La extensión de este tipo de algoritmos da como resultado una arquitectura recurrente. En este caso, las redes recurrentes geométricas se han usado para el reconocimiento de secuencias temporales de líneas. La mayor ventaja de este tipo de red es su flexibilidad en el proceso de entrenamiento dada su naturaleza recurrente. En cuanto a su naturaleza geométrica, este tipo de redes permite trabajar con elementos definidos en álgebra geométrica sin traducirlos a otro lenguaje. El álgebra geométrica ofrece una mayor flexibilidad para la implementación de ideas que contemplan operadores y entidades geométricas dado que éstas se pueden definir en ese mismo lenguaje. La red determina operadores y entidades geométricas en su arquitectura mediante la adaptación de un algoritmo de entrenamiento basado en el gradiente descendente. Esto implica que los pesos de la red ahora tienen un significado geométrico y que la arquitectura de la red permite además tomar en cuenta el factor tiempo. Las redes recurrentes geométricas pueden ser sensibles y adaptadas a las entradas del pasado. No sólo operan en un espacio de entrada, también en un espacio de estado interno- un rastreo de lo que ya ha sido procesado por la red. El espacio de estados permite la representacón (y aprendizaje) de dependencias extendidas temporalmente/secuencialmente sobre invertalos sin especificar (y potencialmente infinitos) de acuerdo a

$$y(t) = G(s(t)) \tag{6.1}$$

$$s(t) = F(s(t-1), x(t)) \tag{6.2}$$

Esta representación puede extenderse en estructuras de redes neuronales artificiales geométricas mediante el álgebra geométrica como se indicó en parte de nuestra investigación.

6.1. Principales resultados

El AG nos ha permitido representar de manera compacta algunos elementos que existen en el medio ambiente y que son considerados en nuestro trabajo como entradas y salidas de nuestros algoritmos. Se representan puntos, líneas y orientaciones de planos principalmente para entrenar a las redes desarrolladas. La arquitectura de las redes neuronales de base radial permite

tener un entrenamiento simple y facilita el aprendizaje mediante el proceso de separación de patrones en la capa oculta. Se utilizó un sistema estereo para adquirir información del ambiente. Además, el brazo robótico también formó parte de nuestros experimentos. Las redes permiten conocer las transformaciones geométricas que existen entre ciertos objetos dados en el espacio 3D. Esto no es posible con redes neuronales convencionales.

Haciendo uso de los trasladores, rotores y motores del álgebra geométrica de motores, se hace más sencilla e intuitiva la representación y la aplicación de cualquier movimiento y transformación existente entre objetos. Estos algoritmos pueden ser usado en tiempo real para conocer la transformación geométrica que existe entre dos conjuntos de orientaciones expresadas como líneas.

La literatura no reporta algún algoritmo similar a alguno que se presenta en este trabajo atendiendo las características de aprendizaje adaptivo y el aspecto geométrico. Este tipo de algoritmos son indudablemente útiles en tareas muy específicas que implican objetivos geométricos relacionados con la representación más fácil e intuitiva de transformaciones, de objetos, líneas y puntos. Como se indicó en la introducción, es fundamental tener algoritmos que puedan tratar con información de cierto tipo específico por razones relacionadas con su desarrollo para la comprensión y modelado de diversos fenómenos. Las RNAs valuadas con reales se desempeñan muy bien en tareas acordes a su tipo de información, asi como las RNVCs. Sin embargo, si se pretende procesar información expresada en un lenguaje de más alto nivel (como el AG) es importante tener algoritmos que puedan tratar con este tipo de información. Hemos aportado dos nuevos algoritmos que pueden tratar con información que expresa varias características geométricas en tiempo real.

Las redes recurrentes nos ayudan a trabajar con información en el tiempo. Por lo tanto, hemos trabajando en el diseño de modelos geométrico básicos basados en el modelo de la red recurrente simple de Elman y nuestra red estática geométrica, para poder tratar con información geométrica que describe las transformaciones cambiantes en el tiempo y las entidades externas a la red. También podríamos aplicar nuestros diseños en diversos problemas cuyas transformaciones puedan ser formuladas con multivectores.

6.1.1. Trabajo futuro

Aún se requiere trabajar en la implementación de nuestros métodos en situaciones donde se requiera una aplicación óptima de nuestros algoritmos. La intregración de la parte mecánica con el software desarrollado es una tarea interesante y debe ser adaptada al problema que se desea considerar. Los métodos descritos pueden ser extendidos, se contempla el caso de una red neuronal estática *MIMO* en la que se consideren al mismo tiempo diversas características de un objeto mediante elementos básicos geométricos.

Nuestro trabajo a futuro implica actualizar el modelo para considerar cualquier tipo de orientación en la secuencia. Para esta actualización, se consideraría también el fundamento teórico de que con suficientes unidades en la red, la longitud de la cadena no tiene límite.

También, se planea actualizar el modelo para considerar varias secuencias de líneas asociadas a un conjunto de elementos geométricos específicos con la misma red.

El concepto de motor se puede extender al concepto de versor que pueda capturar las transformaciones conformales en $\mathcal{G}_{4,1}$ y las afinas en $\mathcal{G}_{6,2}$. De tal manera que puede ser usado en el modelado de redes neuronales geométricas para tratar transformaciones conformales que se dan en diversos problemas de ciencias de la computación e ingeniería. Si se modelan las redes usando el álgebra geométrica conformal $\mathcal{G}_{4,1}$, entonces es posible extender la transformación de rígida a conformal. Esta combinación de conceptos permite que sea posible extraer versores para transformaciones conformales o afinas a partir de la red geométrica, los cuales surgen muy frecuentemente en procesos de situaciones reales donde los movimientos de las líneas evolucionan siguiendo varios tipos de transformaciones.

Bibliografía

[1] Bayro Corrochano E. *Geometric Computing for Perception Action Systems: Concepts, Algorithms, and Scientific Applications*. Springer, 2001.

[2] Ryad Z., Daniel R., and Noureddine Z. The rrbf. dynamic representation of time in radial basis function network. In *Emerging Technologies and Factory Automation, 2001. Proceedings. 2001 8th IEEE International Conference on*, volume 2, pages 737–740, 15 Oct 2001 18 Oct 2001.

[3] Hirose Akira. *Complex-Valued Neural Networks*. Springer, 2006.

[4] Pellionisz A. and Llinàs R. Tensorial approach to the geometry of brain function: cerebellar coordination via a metric tensor. *Neuroscience*, 5:1125–1136, 1980.

[5] Pellionisz A. and Llinàs R. Tensor network theory of the metaorganization of functional geometries in the central nervous system. *Neuroscience*, 16(2):245–273, 1985.

[6] J.J. Koenderink. *The Brain a Geometry Engine. Psychological Research, Vol. 52, pp. 122-127.* Report. Springer, 1990.

[7] Doran C.J.L. *Geometric algebra and its applications to the mathematical physics.* Ph.D. Thesis. University of Cambridge, 1994.

[8] Bayro-Corrochano E. Clifford self-organizing neural network, clifford wavelet network. *Proc. 14th IASTED Int. Conf. Applied Informatics,*, Feb. 20-22, Innsbruck, Austria, pp. 271-274, 1996.

[9] Buchholz S. Bayro-Corrochano E. and Sommer G. Self-organizing clifford neural network. *IEEE ICNN'96 Washington, DC*, June, pp. 120-125, 1996.

[10] Hestenes D. Invariant body kinematics i: Saccadic and compensatory eye movements. *Neural Networks*, 7:65–77, 1993.

[11] Hestenes D. Invariant body kinematics ii: Reaching and neurogeometry. *Neural Networks*, 7:79–88, 1993.

[12] Arena P., Fortuna L., Muscato G., and Xibilia M. G. Multilayer perceptrons to approximate quaternion valued functions. *Neural Networks.*, 10:335–342, 1997.

[13] S. Buchholz and G. Sommer. Quaternionic spinor mlp. In *8th European Symposium on Artificial Neural Networks, ESANN 2000, Bruges*, pages 377–382, 2000.

[14] Banarer V. Perwass Ch. and Sommer G. Spherical decision surfaces using conformal. *Pattern Recognition*, 9:16, 2003.

[15] Isokawa T., Nishimura H., Kamiura N, and Matsui N. Fundamental properties of quaternionic hopfield neural network. In *IJCNN*, pages 218–223, 2006.

[16] Li Mingyu, He Songbai, and Li Xiaodong. Complex radial basis function networks trained by qr-decomposition recursive least square algorithms applied in behavioral modeling of nonlinear power amplifiers. Medical Physics, vol. 32, pp 2371-2379, 2005.

[17] Matsui1 N., Isokawa1 T., Kusamichi1 H., Peper F., and Nishimura H. Quaternion neural network with geometrical operators. *Journal of Intelligent and Fuzzy Systems*, vol. 15, pp 149-164, number 3-4, 2004.

[18] Bayro-Corrochano E., Vallejo R., and Arana-Daniel N. Geometric preprocessing, geometric feedforward neural networks and clifford support vector machines for visual learning. *Neurocomputing*, vol. 67, pp 54-105, 2005.

[19] Tachibana K. and Hitzer E. Tutorial note on ga neural networks. *The 3rd International Conference on Applied Geometric Algebras in Computer Science and Engineering (AGACSE2008)*, Grimma, Germany, 2008.

[20] Kuroe Y. Models of clifford recurrent neural networks and their dynamics. In Proceedings of International Joint Conference on Neural Networks, pages 1035–1041, San Jose, California, USA, July 31-August 5, 2011.

[21] Jaap Suter. Geometric algebra primer. *Reading*, 13(2):157–181, 2003.

[22] Hildenbrand D., Fontijne D., Perwass C., and Dorst L. Geometric algebra and its application to computer graphics. In *Eurographics conference Grenoble*, 2004.

[23] Dorst L., Fontijne D., and Stephen M. *Geometric Algebra for Computer Science: An Object-Oriented Approach to Geometry (The Morgan Kaufmann Series in Computer Graphics)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2007.

[24] Bayro Corrochano E. *Geometric Computing for Perception Action Systems: Concepts, Algorithms, and Scientific Applications*. Springer, 2001.

[25] Hestenes David, Hongbo Li., and Rockwood Alyn. In *Geometric computing with Clifford algebras. Sommer, G. (d.), New algebraic tools for classical geometry*, chapter 1, pages 3–26. Springer-Verlag, London, UK., 2001.

[26] Hestenes D. Old wine in new bottles: A new algebraic framework for computational geometry. geometric algebra with applications in science and engineering. In *Eduardo Bayro Corrochano and Garret Sobczyk (eds.)*, chapter 1, pages 3–17. Birkhauser, New York, 2001.

[27] Ramon y Cajal S. *New Ideas on the Structure of the Nervous System in Man and Vertebrates*. MIT Press, Cambrigde, MA, 1990.

[28] Rojas R. *Neural Networks A Systematic Introduction*. Springer-Verlag, 1996.

[29] Chappelier C. *RST: Une Architecture Connexionniste Pour La Prise En Compte De Relations Spatiales Et Temporelles*. Ecole nationale supérieure des télécommunications, 1996.

[30] Chappelier J. C. and Grumbach A. A kohonen map for temporal sequences. In *Proceedings of NEURAP'95*, pages 104–110, 1996.

[31] Elman Jeffrey L. Finding structure in time. *COGNITIVE SCIENCE*, 14(2):179–211, 1990.

[32] Berthold Intel Michael. A time delay radial basis function network for phoneme recognition. In IEEE Computer Society Press, editor, Proceedings of IEEE International Conference on Neural Networks, volume 7, pages 4470–4473, 1994.

[33] Cybenko G. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2:303–314, 1989.

[34] Rumelhart D. E. and McClelland J. L. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations.* MIT Press, 1986.

[35] Hornik K., Stinchcombe M. B., and White H. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.

[36] Poggio T. and Girosi F. Networks for approximation and learning. In *IEEE Proc.*, volume 78, page 1481:1497, Sept. 1990.

[37] Georgiou G.M. and Koutsougeras C. Complex domain backpropagation. In *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, volume 39, pages 330 – 334, May 1992.

[38] Arena P., Caponetto R., Fortuna L., Muscato G., and Xibilia M.G. Quaternionic multilayer perceptrons for chaotic time series prediction. *IEICE Trans. Fundamentals*, E79-A(October 10):1–6, 1996.

[39] Hamilton W.R. *Lectures on Quaternions.* Springer-Verlag, Heidelberg, 1853.

[40] Pearson J.K. and Bisset D.L. Back propagation in a clifford algebra. *Artificial Neural Networks, 2, I. Aleksander and J. Taylor (Ed.)*, 1992.

[41] Perantonis S. J. and Lisboa P. J. G. Translation, rotation, and scale invariant pattern recognition by high-order neural networks and moment classifiers. *IEEE Transactions on Neural Networks*, 3:241–251, 1992.

[42] Hestenes D. and Sobczyk G. Clifford algebra to geometric calculus: A unified language for mathematics and physics. In *D. Reidel, Dordrecht.* 1984.

[43] Bayro Corrochano E. and Sobczyk G. Applications of lie algebras in the geometric algebra framework. In *Advances in Geometric Algebra with Applications in Science and Engineering. Eduardo Bayro Corrochano and Garret Sobczyk (eds.)*, chapter 13, pages 252–276. Birkhauser, New York, 2000.

[44] Porteous R.I. *Clifford Algebras and the Classical Groups.* Cambridge University Press, Cambridge, 1995.

[45] Haykin S. *Neural Networks: A Comprehensive Foundation.* Prentice Hall PTR, Upper Saddle River, NJ, USA, 2nd edition, 1998.

[46] Eduardo Bayro-Corrochano and Eduardo Vázquez-Santacruz. A geometric radial basis function network for tracking variant 3d transformations. In *IJCNN*, pages 1–6, 2010.

[47] Bernauer E. and Demmou H. Temporal sequence learning with neural networks for process fault detection. In Systems, Man and Cybernetics, 1993. 'Systems Engineering in the Service of Humans', Conference Proceedings., International Conference on, volume 2, pages 375–380, 17-20 Oct 1993.

[48] E. Bayro-Corrochano, K. Daniilidis, and G. Sommer. Hand-eye calibration in terms of motion of lines using geometric algebra. In *In Proc. of the 10th Scandinavian Conference on Image Analysis SCIA'97*, pages 397–404, 1997.

[49] H. Chen. A screw-motion approach to uniqueness analysis of head-eye geometry. In *IEEE Conference on Computer Vision and Pattern Recognition*, Maui, Hawaii, June 3-6, pp. 145-151, 1991.

[50] Lepetit V. and Fua P. Keypoint recognition using randomized trees. volume 28, pages 1465 – 1479, 2006.

Apéndice A

PUBLICACIONES

Revistas

- Eduardo Vazquez-Santacruz, Efraín Castillo Muñiz and Eduardo Bayro Corrochano. "Design of Neural Networks Using Geometric Algebra". Int. Journal of Advances in Applied Clifford Algebras. AACA (Enviado)

Capítulos de Libro

- Eduardo Vazquez-Santacruz and Eduardo Bayro-Corrochano, "A Geometric Radial Basis Function Network for On-line Estimation of Screw Transformations", In Autonomous Robots. Control, Sensing and Perception, Edgar A. Martinez García and Abril Torres Mendez (Eds.), Cuvillier Verlag Göttingen, Germany, Chap. 5, pp.1167-140

Congresos Internacionales

- Eduardo Bayro-Corrochano and Eduardo Vazquez-Santacruz, "A Geometric Radial Basis Function Network for Tracking Variant 3D Transformations", International Joint Conference on Neural Networks. Pages 1-6, July 18-23th, 2010. Barcelona, Spain
- Eduardo Vazquez-Santacruz and Eduardo Bayro-Corrochano, "A Geometric Radial Basis Function Network for Robot Perception and Action", International Conference on Pattern Recognition. Pages 2961-2964, August 23-26th, 2010. Istanbul, Turkey

Eduardo Vazquez-Santacruz and Eduardo Bayro-Corrochano, "A Geometric Radial Basis Function Network for Tracking", 4th Conference on Applied Geometric Algebras in Computer Science and Engineering. June 14-16th, 2010. Amsterdam, Netherlands

- Eduardo Vazquez-Santacruz and Eduardo Bayro-Corrochano, "An Improved Geometric Radial Basis Function Network for Hand-Eye Calibration", International Joint Conference on Neural Networks. Page(s): 1308 1315, July 31, 2011-August 5, 2011. San Jose, California, USA

- E. Ulises Moya-Sanchez and Eduardo Vazquez-Santacruz, "A Geometric Bio-Inspired Model for Recognition of Low-Level Structures", International Conference on Artificial Neural Networks. Pages 429-436, June 14-17th, 2011. Espoo, Finland

- Eduardo Vazquez-Santacruz and Eduardo Bayro-Corrochano, "A Geometric Radial Basis Function Network for Real Time Estimation of Screw Transformations: The Case of Hand-Eye Calibration", 9th Conference on Clifford Algebras and their Applications in Mathematical. ICCA9 2011. July 15-20, 2011. Weimar, Germany

- Eduardo Vazquez-Santacruz and Eduardo Bayro-Corrochano, "A New Geometric Recurrent Neural Network Based On Radial Basis Function and Elman Models", International Joint Conference on Neural Networks. June 10, 2012-June 15, 2012. Brisbane, Australia. (Por aparecer).

CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS DEL I.P.N.
UNIDAD GUADALAJARA

El Jurado designado por la Unidad Guadalajara del Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional aprobó la tesis

Redes neuronales geométricas multicapa y retroalimentadas

del (la) C.

Eduardo Filemón VÁZQUEZ SANTACRUZ

el día 22 de Junio de 2012.

Dr. Eduardo José Bayro Corrochano
Investigador CINVESTAV 3D
CINVESTAV Unidad Guadalajara

Dr. Alexander Georgievich Loukianov
Investigador CINVESTAV 3C
CINVESTAV Unidad Guadalajara

Dr. Mario Angel Siller González Pico
Investigador CINVESTAV 2C
CINVESTAV Unidad Guadalajara

Dra. Susana Ortega Cisneros
Investigador CINVESTAV 2A
CINVESTAV Unidad Guadalajara

Dra. Nancy Guadalupe Arana Daniel
Catedrático
Universidad de Guadalajara

CINVESTAV - IPN
Biblioteca Central
SSIT0011161