UNIVERSIDAD MICHOACANA DE SAN NICOLÁS DE HIDALGO

INSTITUTO DE FÍSICA Y MATEMÁTICAS

Reconocimiento de voz utilizando redes neuronales artificiales

TESIS

QUE PARA OBTENER EL TITULO DE

MAESTRO EN CIENCIAS EN EL ÁREA DE FÍSICA

PRESENTA

LIC. CARLOS EDUARDO LÓPEZ NÚÑEZ

ASESOR

DR. EN CIENCIAS FÍSICAS JOSÉ ANTONIO GONZÁLEZ CERVERA

Morelia, Mich. Febrero de 2015

AGRADECIMIENTOS

Este proyecto de investigación se realizó con la ayuda y apoyo de muchas personas. En primer lugar quiero agradecer a mi asesor, el Dr. José A. González por haber aceptado dirigir este proyecto y ayudarme a resolver las dudas e inquietudes que se generaron en el transcurso del mismo. A los Drs. Francisco S. Guzmán y Umberto Cotti quienes accedieron a realizar la revisión del proyecto y ser sinodales del jurado calificador. También quiero agradecer a Toño, Pancho, Krystal, Ariana y Marcela quienes se prestaron para grabar sus voces y con las cuales se pudieron hacer las pruebas correspondientes dentro del proyecto. Agradecimiento a Mauricio quién me dió consejos y me proporcionó algunos libros sobre los temas aquí expuestos. Por último quiero agradecer a mis papás, hermanos y amigos que siempre me han brindado su apoyo incondicional y a quienes dedico este trabajo.

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

CAPÍTULO 1

INTRODUCCIÓN

En la actualidad usamos distintos algoritmos computacionales para resolver una amplia variedad de problemas tanto en física, matemáticas, biología, en ingenierías o en cualquier otra rama de la ciencia en la que se necesitan hacer una gran cantidad de cálculos aritméticos o que no es posible realizarlos analíticamente, por lo que es necesario el uso de métodos numéricos. Problemas en los que se requiere resolver ecuaciones diferenciales ordinarias o parciales, se pueden resolver con el método de Runge-Kutta para el primer caso o el método de diferencias finitas para el segundo caso [Chapra and Canale, 2007]. Problemas de optimización o de inteligencia artificial se resuelven utilizando algoritmos genéticos [Popa, 2012]. El reconocimiento de voz se puede realizar a través de algoritmos que resuelven los modelos ocultos de Márkov [Dymarski, 2001]. Así podríamos seguir enunciando una lista de problemas, pero todos los problemas antes mencionados tienen algo en común: pueden ser resueltos utilizando redes neuronales artificiales.

Las redes neuronales artificiales (o RNA) son una herramienta computacional que basa su funcionamiento en el comportamiento biológico de las redes neuronales. Al igual que nuestro cerebro se va adaptando en base a la experiencia que adquiere con el tiempo para responder a la información que percibe, las RNA procesan los datos que se introducen en ellas en base a ejemplos previos, los cuales adaptan a la red para obtener ciertos resultados de acuerdo a lo que se quiere resolver.

Las aplicaciones de las redes neuronales artificiales son muy variadas, como lo son el reconocimiento de voz, la conversión de texto a voz, la detección de enfermedades, clasificación de galaxias, entre muchas otras [Kroese and Van der Smagt, 1996]. Una de las principales dificultades al usar una RNA, es encontrar una manera adecuada de codificar la información que se quiere introducir en la red para su procesamiento. Esto ya dependerá del problema que se quiere solucionar.

A pesar de que existen diferentes áreas de la ciencia en donde se pueden emplear las RNA, nosotros estamos interesados en resolver problemas relacionados con sistemas físicos. En este primer acercamiento que tendremos con las RNA se optó por realizar análisis de audio, ya que el tema es de mi interés y deseo obtener mayor conocimiento en lo referente a ondas de sonido y el procesamiento de las mismas al digitalizarlas para su análisis. En particular se quiere extraer información relevante de muestras de audio que contengan palabras de diferentes personas, para su reconocimiento. Con esto nos referimos al echo de que al introducir la información dentro de la RNA, esta nos indique a que persona pertenece la voz del grupo de personas que se tiene para el estudio. Como hemos mencionado, necesitaremos una forma de caracterizar la información que se procesará, por lo que se necesitará hacer uso de herramientas matemáticas, como lo son las transformadas de Fourier y mas específicamente las transformadas discretas de Fourier, ya que como veremos mas adelante, al trabajar con senãles digitales tendremos que trabajar con valores discretos. Esta herramienta matemática nos facilitará el estudio para obtener la información relevante de las muestras de audio que se introducirán en la red.

Por otro lado en este proyecto se desea conocer como afectan ciertos parámetros dentro de las redes al modificarlos y de los cuales hablaremos y analizaremos en los próximos capítulos. Estos parámetros se explorarán usándolos para resolver distintos problemas aquí planteados, con los que primeramente se verificará el funcionamiento de la red, para posteriormente entrar de lleno en el estudio de las señales digitales de audio que contienen las voces. Todo esto con el fin de obtener conocimientos que nos sirvan en el futuro para emplear las RNA de manera mas eficiente en problemas físicos mas complejos.

Tomando todo esto en consideración deseamos utilizar un tipo de estructura de RNA específico (ya que existen una gran variedad de estructuras que se pueden utilizar y que tiene diferentes usos) para el reconocimiento de voz, debido a que las técnicas tradicionales para el análisis de audio pueden emplear demasiado tiempo para procesar la información. En las RNA se pueden usar técnicas para paralelizar el código y procesar bloques de datos al mismo tiempo y con esto optimizar el tiempo de cálculo. También tiene la ventaja de procesar datos que incluyen ruido generado al momento en que se guardaron los mismos, es este caso al grabar los diferentes patrones de sonidos generados por la persona que habla, intervienen condiciones del lugar donde se esta grabando, como las dimensiones del lugar, el material del que esta hecho las paredes que lo delimitan (si es que existen), la temperatura del aire en ese momento, etc. y que pueden producir cambios en la señal grabada. Si se quisiera hacer reconocimiento de voz comparando los datos guardados, uno por uno (cada elemento de la lista que contiene la señal de audio, que usualmente es del orden de 44000 elementos por segundo grabado), con las muestras del audio que se desea analizar, resultaría muy difícil, ya que cualquier cambio que sufra la información complicaría el estudio. Este tipo de inconvenientes se pueden arreglar al usar una red neuronal artificial, debido a que se le dan ciertos patrones de muestra que incluyan diferentes condiciones, en el caso de la grabación de voz, grabaciones con diferentes temperaturas ambiente o de diferentes dimensiones en el lugar para que la red sea capaz de procesar la información cuando existan este tipo de variaciones y aún cuando los patrones que se quieren comparar, difieran de los que se tenían de muestra inicialmente. Esta característica es la que hace que una red generalice para una gran cantidad de problemas, utilizando patrones de muestra concretos para adaptar a la red y encontrar una solución a lo planteado.

En este primer capítulo unicamente presentamos las motivaciones para realizar este proyecto y la idea general del problema que se quiere resolver.

El capítulo dos tratará las ideas básicas sobre el funcionamiento de las neuronas biológicas para poder hacer su analogía mediante un modelo matemático que las describa. Se hablará sobre los diferentes parámetros que constituyen estas redes y de como imple-

mentar el algoritmo.

El tercer capítulo contiene información de como es que se puede hacer uso de una señal análoga, convirtiéndola en una señal digital para su manipulación y procesamiento. Se estudiarán las transformadas discretas de Fourier para emplearlas en el análisis de las muestras de audio grabadas previamente.

En el capítulo cuatro se plantean problemas en los que se utilizarán RNA para su solución. Se estudia el funcionamiento de la red al modificar los diferentes parámetros que la caracterizan y se hace uso de las transformadas discretas de Fourier sobre las muestras de audio grabadas, seleccionando información esencial, para posteriormente introducirlas dentro de la red y poder clasificar la información de las voces, dependiendo de la persona a la que pertenece la voz.

Por último en el capítulo cinco se muestran las conclusiones a las que se llegaron con este proyecto de investigación y el trabajo que se desea desarrollar en el futuro utilizando las RNA.

Espero que este trabajo sirva para mostrar el uso que se les puede dar a las RNA y que despierte interés y curiosidad en el lector para que se adentre en el mundo de las redes neuronales artificiales. Comencemos entonces con lo que son las RNA, su estructura y sus características.

CAPÍTULO 2

REDES NEURONALES ARTIFICIALES

Una red neuronal artificial es una herramienta para procesar información la cual se desarrolló a mediados del siglo XX teniendo en cuenta como es que funcionan las neuronas biológicas en el interior de nuestro cerebro, las cuales estan interconectadas entre si para recibir, procesar y transmitir dicha información formando una gran red y por la cual recibe su nombre. Todo esto se realizó partiendo de la idea del procesamiento en paralelo, en el que las unidades que procesan la información trabajan al mismo tiempo bloques de datos. A pesar de la amplia variedad de aplicaciones que tienen las RNA, hubo un tiempo en el que se frenó el impulso que estaban teniendo debido a algunos resultados que se encontraron con el primer tipo de estructura que se desarrolló llamado perceptrón [Graupe, 2007] y quedaron muy pocas personas que continuaron las investigaciones en esta área. Afortunadamente después se desarrollaron otro tipo de estructuras en las redes como la RNA backpopagation (de la que hablaremos mas a detalle en las próximas secciones), la cual resolvió algunas limitaciones de las redes perceptrón y fue lo que ayudó a que se retomara el estudio de las RNA. Hoy en día son una herramienta muy utlizada y su uso va en aumento.

2.1 Estructura biológica de una neurona

Para que se pudieran desarrollar las RNA a como se implementan hoy en día, fue necesario primero que hubieran avances en el área de la neurobiología en cuanto a la forma en que se sabe (aunque de forma muy simplificada) como es que se comunican las neuronas unas con otras y como es que se procesa la información dentro de las mismas. Una neurona se puede considerar que esta formada esencialmente de tres partes: el núcleo, las dendritas y el axón (Figura 2.1). En el núcleo se procesa toda la información que recibe proveniente de otras neuronas y puede ser de distintos tipos, dependiendo del sentido del que viene, ya sea algo que vemos, escuchamos, olemos, sentimos o degustamos. Usando en conjunto esta información generada por nuestros sentidos y en base a la experiencia antes obtenida, nuestras neuronas guardan información y al momento de requerirlo la usan para generar una respuesta a algún estímulo externo.

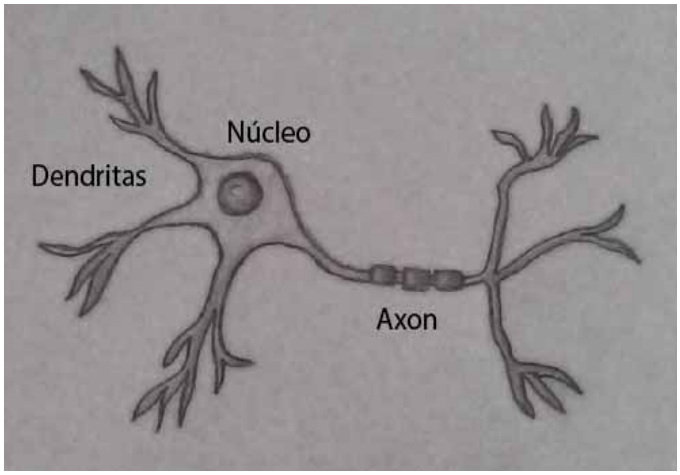

Figura 2.1: Partes principales de una neurona que se encargan de recibir, procesar y transmitir la información dentro de la red neuronal.

Esta información es transmitida a través de un proceso electroquímico. La membrana celular separa el plasma intracelular del fluido intersticial externo a la célula. Dentro y fuera de la membrana existen iones (potasio, sodio, cloro y iones orgánicos) los cuales generan una diferencia de potencial, llamado potencial de reposo (mas detalles en [Freeman and Skapura, 1991]).

Las conexiones con otras neuronas ocurren en distintos puntos en las dendritas, las cuales modifican el potencial en reposo. Estos potenciales de entrada alteran a la célula, ya sean exitándola (disminuyendo la polarización de la célula) o inhibiéndola (aumentando la polarización). Los potenciales de entrada se suman en el axón y si la despolarización supera un cierto umbral, se genera un potencial de acción el cual viaja a otras neuronas.

Así es como se envía la información a otras neuronas y el proceso se repite en las neuronas a las que llega. A continuación se estudiará la analogía en una RNA al recrear este proceso matemáticamente.

2.2 Estructura de una RNA

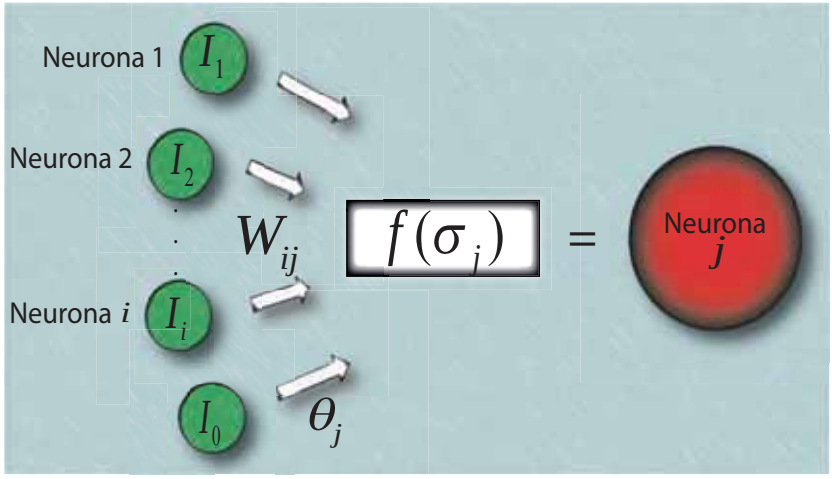

Figura 2.2: Neurona j-ésima activada al recibir la información proveniente de las neuronas I_i.

Las RNA se forman con varias neuronas o elementos procesadores (EP) que simulan el comportamiento de las neuronas biológicas. Cada neurona funciona como se muestra en la Figura 2.2, en donde recibe la información que proviene de otros EP con los que esta conectado, en este ejemplo los elementos son I_i. Las conexiones entre las neuronas se simulan mediante los números W_{ij} a los que se les llama peso y que nos indican que

tan fuertemente están conectados estos elementos entre si y que sería el análogo a las diferencias de potencial que hay en las dendritas en el caso de la neurona biológica. Si el número W_{ij} es positivo, nos indica que la conexión que va del EP I_i al elemento j es del tipo que estimula a la neurona; si el número W_{ij} es negativo, quiere decir que el elemento I_i inhibe al EP j y por ultimo, si W_{ij} es cero quiere decir que no hay conexión entre estos elementos.

Aquí estamos mostrando el esquema en donde las neuronas están conectadas mediante estos números llamados pesos, pero el primer tipo de unidades o elementos procesadores (unidades McCulloch-Pitts) que se crearon no tenían conexiones y se pueden usar para crear compuertas lógicas con álgebra booleana [Rojas, 1996].
El valor que tendrá la neurona j estará relacionado con el valor que tienen las neuronas que envían la información y como son las conexiones entre estas. Una manera sencilla en la que se puede pensar, es tomar en cuenta estas contribuciones como una combinación lineal

$$W_{1j}I_1 + W_{2j}I_2 + \ldots + W_{ij}I_i = \sum_{1}^{i} W_{ij}I_i = \sigma_j \tag{2.1}$$

en donde la i corre desde 1, hasta el número de neuronas con las que existe una conexión, para obtener así obtener la contribución total, nombrándose a este tipo de unidades con esta regla de propagación, unidades sigma. En ocasiones se suele introducir también un término constante (θ_j) llamado bias

$$\sigma_j = \sum_{1}^{i} W_{ij}I_i + \theta_j \tag{2.2}$$

Este término también se puede ver como la contribución de una neurona con valor de 1 (I_0) multiplicado por un peso igual al valor de θ_j ($W_{0j}I_0 = \theta_j$), por lo que se puede pensar que se tiene la Ecuación 2.1 pero la suma empezando desde cero. Existen otro tipo de reglas de propagación menos usadas como las unidades sigma-pi introducidas por Feldmand y Ballard [Feldmand and Ballard, 1982], pero por el momento solo se usarán unidades sigma por simplicidad.
Una vez que tenemos como introduciremos la información que procesará la neurona j, necesitamos saber que regla o transformación se aplicarán a estos datos para generar el valor de salida. Una RNA puede ser vista como una máquina para aproximar funciones, transformando vectores de dimensión m en vectores de dimensión n, mediante una

combinación de funciones primitivas. En el esquema anterior equivaldría a aproximar una función $f : \Re^i \to \Re$, es decir, una función que toma los valores de las neuronas I_i y las transforma en el valor que produce la neurona j.
Consideremos la aproximación de una función (infinitamente diferenciable y con todas sus derivadas contínuas) alrededor de un punto a por una serie de Taylor

$$f(x) = f(a) + \frac{(x-a)}{1!}f'(a) + \frac{(x-a)^2}{2!}f''(a) + \ldots + \frac{(x-a)^n}{n!}f^n(a) + \ldots \qquad (2.3)$$

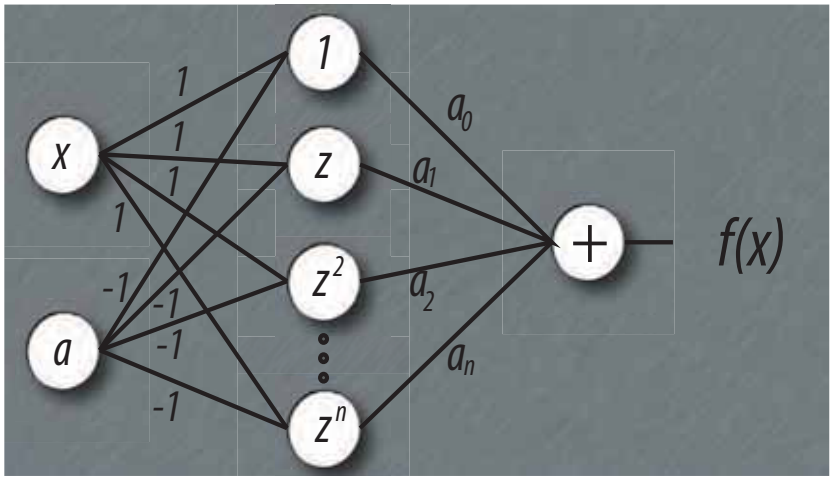

Figura 2.3: RNA que aproxima una función $f(x)$ a través de las funciones $1, z, z^2, \ldots, z^n$ en las neuronas donde llega la información de entrada (x y a) mediante los pesos constantes -1 y 1. La información después es enviada a una última neurona (usando los pesos $c_n = \frac{f^n(a)}{n!}$) que tiene como función sumar todos los valores que le llegan.

Este tipo de aproximación puede ser modelada mediante una red neuronal artificial que tenga la estructura de la Figura 2.3, en la que vemos como al establecer ciertos valores en los pesos y las funciones adecuadas en las neuronas, podemos obtener el mismo resultado de aproximar por una serie de Taylor. Solo hay que calcular los pesos C_n, que en este caso podría ser analíticamente derivando la función las veces necesarias y evaluándola en el punto a. En el caso de las redes neuronales artificiales estos pesos se encuentran mediante el uso de un algoritmo de aprendizaje, del cual hablaremos más adelante. La principal diferencia entre aproximar por una serie de Taylor o

por una RNA, es que, la función f no se tiene explícitamente, si no implícitamente mediante un conjunto de ejemplos de entrada-salida. Conocemos f solo en algunos puntos pero queremos adaptar los parámetros de la red de manera óptima para reflejar la información conocida y extrapolar esta información a un conjunto nuevo de datos.

En el esquema antes visto la información de entrada o los valores donde se conoce la función corresponden a las neuronas I_i y la salida es el valor que produce la neurona j al utilizar la función f para transformar σ_i. Esta función se denomina función de activación, la cual activará el valor de salida que tendrá la neurona, depués de procesar la información con tal función y que posteriormente se transmitirá a otras neuronas. Esto es análogo a cuando los potenciales de entrada superan el umbral que hace que se dispare el potencial de activación en la neurona biológica. Esta función de activación varía dependiendo del tipo de datos que se introducen y del tipo de problema que se quiere resolver. Las funciones de activación mas comunes son las que se muestran en las Figuras 2.4, 2.5 y 2.6.

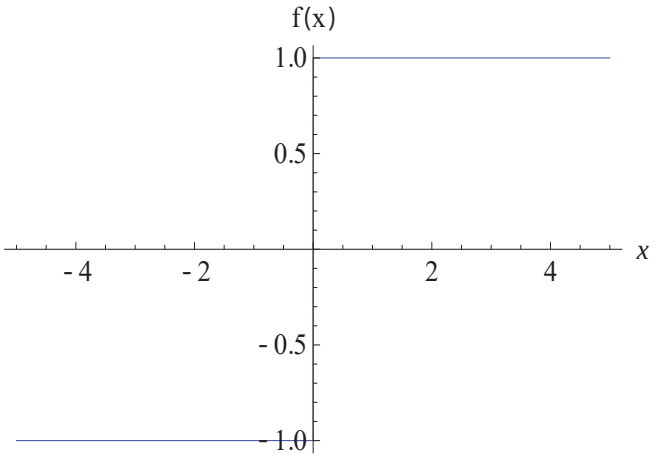

Figura 2.4: Función escalón $f(x) = \begin{cases} -1 & si \ x < 0, \\ 1 & si \ x > 0 \end{cases}$

Dentro de las RNA se pueden distinguir 3 tipos de unidades: las unidades de entrada que reciben información externa a la la red, las unidades de salida que envían información fuera de la red y las unidades ocultas cuyas señales de entrada y salida

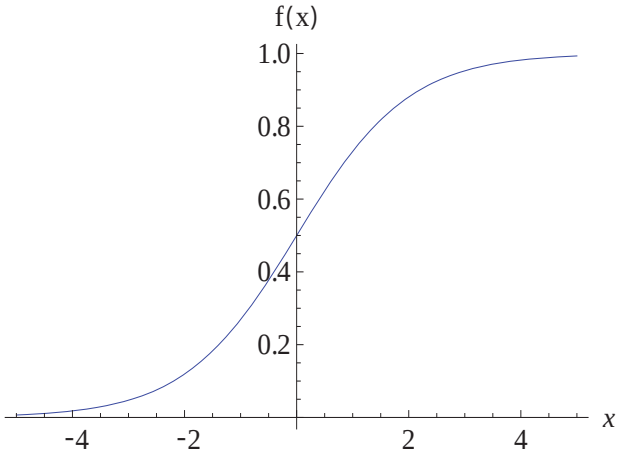

Figura 2.5: Función sigmoide $f(x) = \frac{1}{1+Exp[-x]}$

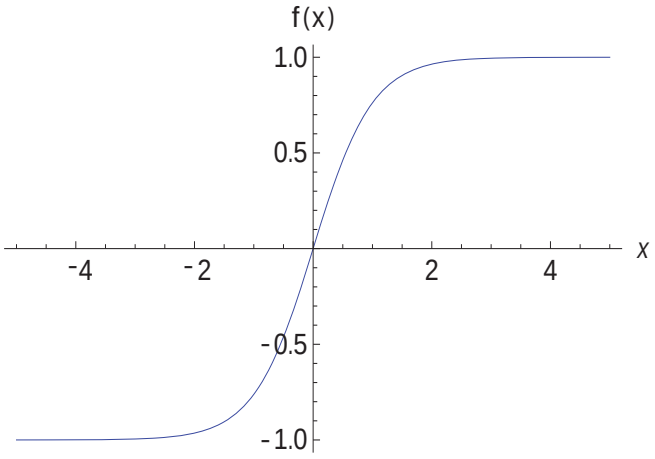

Figura 2.6: Función tangente hiperbólica $f(x) = \frac{Exp[x]-Exp[-x]}{Exp[x]+Exp[-x]}$

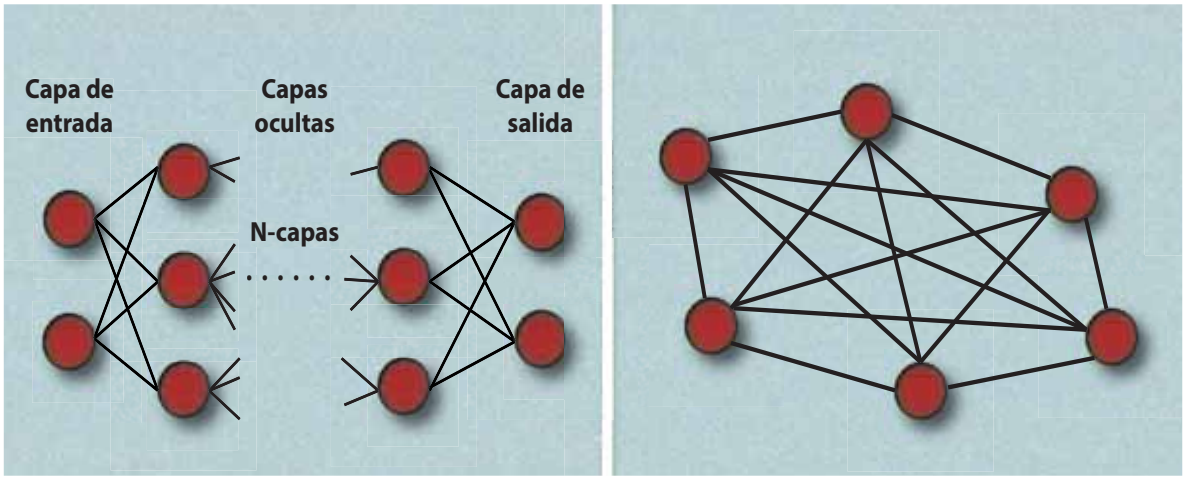

Figura 2.7: A la izquierda la estructura general de una RNA alimentada hacia adelante, donde se procesa la información por capas. A la derecha una RNA recurrente, donde la información puede ser procesada por una misma neurona varias veces o por ciclos.

permanecen dentro de la red. Con esta distinción podemos clasificar las topologías de las RNA de dos maneras

Redes con alimentación hacia adelante . En este tipo de redes la información fluye en una sola dirección, que es desde las unidades de entrada, hacia las unidades de salida. Las unidades de entrada, así como las de salida y las ocultas, estan estructuradas por capas en donde la información no puede trasmitirse hacia atrás, es decir, que las unidades ocultas transmitan la información hacia las unidades de entrada. Tampoco esta permitido que existan conexiones entre neuronas de la misma capa. Como no existen ciclos, es decir, cada unidad calcula un valor que después es transmitido a otras unidades en las capas siguientes, se asume que los cálculos se realizan sin retrasos, por lo que no es necesario una sincronización de los EP.

Redes recurrentes . Este tipo de estructura permite que existan conexiones entre cualquier neurona dentro de la red por lo que pueden existir ciclos entre los EP, es decir, que la información que producen puede ser usada otra vez como entrada

para realizar un nuevo cálculo, por lo que aquí si es importante considerar el tiempo de procesamiento en cada EP.

Las RNA están compuestas de varios EP o neuronas y existen varios tipos de redes dependiendo de como es que están conectadas las neuronas, la forma en que se entrena, la función de activación y los datos que se manejan.
La estructura general básica de una RNA con alimentación hacia adelante se puede observar en la Figura 2.7 (izquierda). En primer lugar se tiene la capa con las unidades de entrada que sólo recibirán la información sin procesarla y la transmitirán hacia neuronas en la siguiente capa. Posteriormente puede haber una o más capas de unidades ocultas (llamadas capas ocultas) que procesarán la información y las mandarán a la capa con las unidades de salida. Si la red solo cuenta con una capa de unidades de entrada y una capa de unidades de salida, se dice que la red es de una capa (debido que las unidades de salida son las únicas que realizan cálculos). Si la red tiene una o más capas ocultas, se dice que la red es multicapa. El número de capas ocultas puede variar dependiendo del problema, pero generalmente se prefiere solo utilizar una capa, ya que para fines prácticos, puede ser mejor tener una sola capa oculta con un mayor número de unidades, a tener varias capas ocultas con menor número de unidades por capa, debido a que el número de cálculos a realizar se incrementa considerablemente. Por último se tiene la capa con las unidades de salida que son las que reciben toda la información procesada proveniente de las unidades en las capas previas. No se entrará en detalle por ahora sobre las redes recurrentes, ya que la estructura que nos interesa de momento, son las que tienen alimentación hacia adelante.

Los seres humanos aprendemos en base a experiencias que tenemos en nuestra vida cotidiana y con el conocimiento que adquirimos. Cuando sufrimos por primera vez una quemadura, causada por fuego por mencionar un ejemplo, lo que nuestro cerebro percibe en ese momento es la información proveniente de los sentidos involucrados en captar tal experiencia, para así poder generar una respuesta al estímulo recibido. En este caso procede a indicar que siente cierto grado de dolor y se nos queda guardada esta experiencia para el futuro. Supongamos que en este caso, la quemadura se ocasionó con la lumbre de una estufa, por lo que si en el futuro vemos fuego en una fogata, sabremos que si nos acercamos nos podemos quemar, aunque no sean exactamente las mismas

condiciones, pero podemos detectar el peligro debido a la información que obtuvimos previamente.
Algo similar sucede al momento de emplear una RNA. Nosotros le indicaremos a la red los resultados que queremos obtener en las unidades de salida al seleccionar un conjunto de datos que introduciremos en las unidades de entrada y la red adaptará los pesos mediante alguna regla, tal que al procesar los datos que introducimos, nos produzca los resultados que queremos. A este procedimiento se le llama "entrenamiento de la red" y entre mas conjuntos de datos entrenemos, mas eficiente será una vez que esté entrenada para obtener los resultados deseados, aún cuando los datos que introducimos varíen un poco respecto a los que usamos para entrenar a la red. Esta característica es la que le brinda a la red la flexibilidad de ser aplicada a un gran número de problemas.

Existen dos maneras de entrenar una RNA:

Entrenamiento supervisado en el que se introducen a la red conjuntos de datos en las unidades de entrada y una vez que se propaga esta información a las unidades de salida, se comparan con los objetivos propuestos, a través de una persona que proporciona estos datos, mide el error y utiliza un algoritmo para corregir los pesos de manera que este error se minimice.

Entrenamiento sin supervisión en el que al introducir un conjunto de datos, el resultado numérico es desconocido, por lo que el sistema debe descubrir características estadísticamente mas sobresalientes de la población de datos de entrada para poder adaptar la red.

Una vez que se escoge una de estas dos formas en que se va a entrenar a la red, se necesita una regla de aprendizaje para adaptar los pesos de manera que se obtengan los objetivos deseados y dependerá del tipo de red que se usa.

Aunque se mostró la estructura general de una red alimentada hacia adelante y los parámetros básicos que se requieren para desarrollar una RNA, en la práctica varían un poco en base a lo que necesitemos. Por ejemplo, la primer red neuronal que se desarrolló fué implementada por Frank Rosenblatt, llamada perceptrón [Rosenblatt, 1958] y

se usa principalmente para clasificación de patrones. La estructura SOM (Self Organizing Map o mapa auto-organizado en español) [Kohonen, 1988] que en vez de tener una capa oculta unidimensional, tiene una bidimensional y utiliza entrenamiento sin supervisión, o la red tipo Backpropagation (o retroalimentada en español) que es una generalización del perceptrón. Existen incluso redes que utilizan estructuras de varios tipos, tal es el caso de la red Counterpropagation [Freeman and Skapura, 1991], la cual es una combinación de una red tipo SOM y otra Backpropagation.
Estas son sólo algunas de las estructuras que existen y que solo mencionaremos por el momento, ya que como hemos dicho anteriormente, el tipo de red que se utiliza, así como otros factores como pueden ser la regla de propagación o el tipo de unidades, dependen ampliamente de lo que se quiere solucionar.

Nosotros nos enfocaremos en estudiar las redes tipo Backpropagation, por que tienen una implementación relativamente sencilla y la variedad de aplicaciones en donde se puede emplear es muy grande y como ya se dijo, generalizan las redes tipo perceptrón, las cuales por si solas ya se pueden utilizar para resolver ecuaciones diferenciales ordinarias y parciales [Lagaris et al., 1997b], que a su vez pueden resolver problemas como la ecuación de Schr¨odinger o la de Dirac [Lagaris et al., 1997a]. Pasemos entonces a estudiar las características y propiedades de este tipo de redes.

2.3 RNA tipo Backpropagation

En esta sección estudiaremos el funcionamiento de una RNA Backpropagation o con retroalimentación. Se llama así por el hecho de que una vez que se propaga la información desde las unidades de entrada a las de salida, como es usual en las redes alimentadas hacia adelante, se van actualizando los pesos desde las unidades de salida a las de entrada, es decir, en dirección opuesta a como se propagó inicialmente la información. Utilizaremos el esquema de la sección anterior (Figura 2.2) para el procesamiento de la información, solo que en lugar de tener que la información solo llega a una neurona (en aquel caso la neurona j), tendremos que la información es recibida por varias neuronas a la vez. Nosotros usaremos una capa de unidades ocultas a las que nombraremos h_j, en donde j corre desde uno hasta el número de neuronas que

asignemos en tal capa. Así se tendrá entonces, que la información producida por las neuronas que reciben la información de las unidades de entrada es

$$h_j = f(\sigma_j) = f(\sum_i WI_{ij} I_i + \theta_j) \tag{2.4}$$

donde I_i son las neuronas de entrada, WI es una matriz de i x j que contiene cada uno de los pesos o conexiones entre las unidades de entrada y la capa oculta, θ_j es el bias asociado a la neurona h_j y f es la función de activación.

Similarmente el valor que producirán las neuronas en la capa de salida (llamadas s_k, con k el número de neuronas en esta capa) tomando la información proveniente de las neuronas en la capa oculta es

$$s_k = f(\sigma_k) = f(\sum_j WH_{jk} h_j + \theta'_k) \tag{2.5}$$

con WH una matriz de j x k que contiene el valor de los pesos entre las neuronas de la capa oculta y las de salida.

Por último necesitaremos un vector del mismo tamaño que la capa de salida, en donde asignaremos ciertos valores que deseamos obtener después de que se propague la información de entrada y a los que llamaremos objetivos (t_k). Estos objetivos nos servirán para identificar o clasificar la información de alguna manera que nos convenga.

La estructura que resulta se observa en la Figura 2.8.

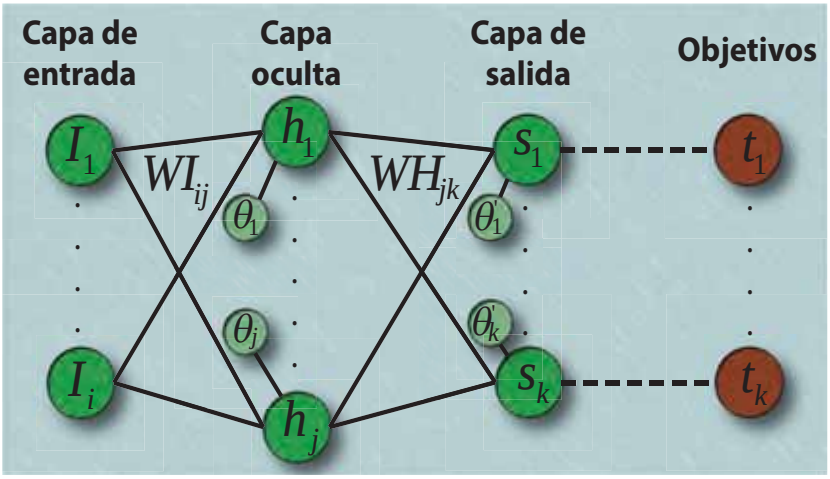

Figura 2.8: Estructura de una RNA tipo backpropagation con una capa oculta y con la notación utilizada.

Al inicializar la red no conocemos nada sobre las neuronas en las capas ocultas ni de los pesos entre las capas, por lo que se pueden inicializar con valores arbitrarios. Necesitamos ahora de una regla de aprendizaje, con la cual se actualizarán los pesos de tal manera que cuando se propague la información nos resulten los valores de los objetivos propuestos en la capa de salida. En general, al propagar la información de entrada y obtener los valores de salida, se tendrán valores diferentes a los deseados ($s_k \neq t_k$). El error promedio o función de error entre los valores deseados y los obtenidos en las unidades de salida, se toma como el error cuadrático medio

$$E = \sum_{l=1}^{k} e_l = \sum_{l=1}^{k} \frac{1}{2}(t_l - s_l)^2 \tag{2.6}$$

Para hacer que los valores de salida que produce la red se aproximen a los valores que se quieren obtener, se necesitan de varios pasos o iteraciones en los que la red propagará la información y adaptará los pesos para que cada vez se acerquen más a los valores de los objetivos propuestos. A este proceso de adaptación de los pesos es al que se conoce como regla de aprendizaje. Lo que se usa en la redes tipo backpropagation para adaptar los pesos es la idea de minimizar la función de error que dependerá de estos valores (ya que los únicos parámetros libres son los los pesos), mediante la idea del gradiente. El gradiente de una función escalar, da como resultado la dirección máxima donde cambia dicha función, por lo que si tomamos la dirección opuesta del gradiente, obtendremos la dirección en la que la función se minimiza más rápido. Por esta razón necesitamos trabajar con funciones diferenciables y en este caso se usan funciones de activación como la sigmoide o la tangente hiperbólica, debido a que el error depende de los valores s_k que a su vez se obtienen al aplicar la función de activación.
El gradiente de la función error que depende de n variables $E((w_1), (w_2), \ldots, (w_n))$ se define como el vector

$$\nabla E = \big(\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \ldots, \frac{\partial E}{\partial w_n}\big)$$

Si consideramos los pesos entre la capa oculta y la de salida se tendrán entonces j x k variables y para cada una de estas direcciones tomando el sentido opuesto, se tienen entonces expresiones de la forma

$$\Delta WH_{jk} \approx -\frac{\partial E}{\partial WH_{jk}}$$

donde ΔWH_{jk} es igual a $WH_{jk}^{nuevo} - WH_{jk}^{viejo}$ (nuevo es el peso adaptado y viejo es el peso sin adaptar). Si queremos que se cumpla la igualdad se introduce una constante de proporcionalidad γ a la que se llama constante de aprendizaje:

$$\Delta WH_{jk} = -\gamma \frac{\partial E}{\partial WH_{jk}} \tag{2.7}$$

Esta regla de aprendizaje es conocida como la regla delta generalizada.

Se trabajará primero con el término de la derivada con respecto al error y al final se sustituirá el valor en la ecuación anterior. Los términos que dependen de WH_{jk} son únicamente los s_k, por lo que debemos aplicar regla de la cadena

$$\frac{\partial E}{\partial WH_{jk}} = \frac{\partial}{\partial WH_{jk}} \sum_{l=1}^{K} e_l = \frac{\partial}{\partial WH_{jk}} \sum_{l=1}^{k} \frac{1}{2}(t_l - s_l)^2 = -\sum_{l=1}^{K} \frac{1}{2} 2(t_l - s_l) \frac{\partial s_l}{\partial WH_{jk}}$$

donde K es el número de neuronas en la capa de salida para no confundirlo con el subíndice k (lo mismo para I y J que usaremos mas adelante). Sustituyendo el valor de s_l usando la Ecuación 2.5

$$\begin{aligned}
\frac{\partial E}{\partial WH_{jk}} &= -\sum_{l=1}^{K}(t_l - s_l)\frac{\partial}{\partial WH_{jk}} F(\sum_{m=1}^{J} WH_{ml}h_m) = -\sum_{l=1}^{K}(t_l - s_l)\frac{\partial}{\partial WH_{jk}} F(\sigma_l) \\
&= -\sum_{l=1}^{K}(t_l - s_l)\frac{\partial F(\sigma_l)}{\partial \sigma_l}\frac{\partial \sigma_l}{\partial WH_{jk}} = -\sum_{l=1}^{K}(t_l - s_l)F^{'}(\sigma_l)\frac{\partial}{\partial WH_{jk}}\sum_{m=1}^{J} WH_{ml}h_m \\
&= -\sum_{l=1}^{K}(t_l - s_l)F^{'}(\sigma_l)\sum_{m=1}^{J}\frac{\partial WH_{ml}}{\partial WH_{jk}}h_m = -\sum_{l=1}^{K}(t_l - s_l)F^{'}(\sigma_l)\sum_{m=1}^{J}\delta_{mj}\delta_{lk}h_m \\
&= -\sum_{l=1}^{K}(t_l - s_l)F^{'}(\sigma_l)\delta_{lk}h_j = -(t_k - s_k)F^{'}(\sigma_k)h_j
\end{aligned} \tag{2.8}$$

este resultado lo sustituimos en la Ecuación 2.7 para llegar a

$$WH_{jk}^{nuevo} = WH_{jk}^{viejo} + \gamma(t_k - s_k)F^{'}(\sigma_k)h_j \tag{2.9}$$

Adaptemos ahora los pesos WI_{ij} que van de la capa de entrada a la capa oculta, ya que como se mencionó anteriormente, el proceso de adaptación va en sentido opuesto a la que se propagó la información. El procedimiento es similar al que se acaba de utilizar solo que ahora se deriva con respecto a WI_{ij} y donde aparezcan los términos

de la matriz de pesos WH se usarán los que se adaptaron en el paso anterior

$$\begin{aligned}\frac{\partial E}{\partial WI_{ij}} &= \frac{\partial}{\partial WI_{ij}}\sum_{l=1}^{K}\frac{1}{2}(t_l - s_l)^2 = -\sum_{l=1}^{K}\frac{1}{2}2(t_l - s_l)\frac{\partial s_l}{\partial WI_{ij}} = -\sum_{l=1}^{K}(t_l - s_l)\frac{\partial}{\partial WI_{ij}}F(\sigma_l) \\ &= -\sum_{l=1}^{K}(t_l - s_l)F^{'}(\sigma_l)\frac{\partial \sigma_l}{\partial WI_{ij}} = -\sum_{l=1}^{K}(t_l - s_l)F^{'}(\sigma_l)\sum_{m=1}^{J}\frac{\partial}{\partial WI_{ij}}(WH_{ml}h_m)\end{aligned} \tag{2.10}$$

sustituimos el valor de h_m usando la Ecuación 2.4 y como estamos derivando con respecto a WI_{ij}, entonces WH_{ml} sale constante

$$\begin{aligned}\frac{\partial E}{\partial WI_{ij}} &= -\sum_{l=1}^{K}(t_l - s_l)F^{'}(\sigma_l)\sum_{m=1}^{J}WH_{ml}\frac{\partial}{\partial WI_{ij}}F(\sigma_m) \\ &= -\sum_{l=1}^{K}(t_l - s_l)F^{'}(\sigma_l)\sum_{m=1}^{J}WH_{ml}F^{'}(\sigma_m)\frac{\partial}{\partial WI_{ij}}\sum_{n=1}^{I}WI_{nm}I_n \\ &= -\sum_{l=1}^{K}(t_l - s_l)F^{'}(\sigma_l)\sum_{m=1}^{J}WH_{ml}F^{'}(\sigma_m)\sum_{n=1}^{I}\delta_{in}\delta_{jm}I_n \\ &= -\sum_{l=1}^{K}(t_l - s_l)F^{'}(\sigma_l)\sum_{m=1}^{J}WH_{ml}F^{'}(\sigma_m)\delta_{jm}I_i \\ &= -\sum_{l=1}^{K}(t_l - s_l)F^{'}(\sigma_l)WH_{jl}F^{'}(\sigma_j)I_i\end{aligned} \tag{2.11}$$

Por lo que llegamos a la expresión

$$\frac{\partial E}{\partial WI_{ij}} = -F^{'}(\sigma_j)I_i\Big[\sum_{l=1}^{K}(t_l - s_l)F^{'}(\sigma_l)WH_{jl}\Big] \tag{2.12}$$

utilizando nuevamente la Ecuación 2.7 pero en vez de los subíndices jk usamos ij

$$WI_{ij}^{nuevo} = WI_{ij}^{viejo} + \gamma F^{'}(\sigma_j)I_i\Big[\sum_{l=1}^{K}(t_l - s_l)F^{'}(\sigma_l)WH_{jl}\Big] \tag{2.13}$$

Por ahora se utilizó una red con una capa oculta pero el procedimiento para adaptar los pesos si se tienen mas capas ocultas es similar al que aquí mostramos.

Si tomamos la función sigmoide como función de activación, al derivarla obtenemos

$$f(x) = \frac{1}{1 + Exp[-x]} \Rightarrow f'(x) = \frac{Exp[-x]}{(1 + Exp[-x])^2} = [1 - f(x)]f(x)$$

al sustituir este resultado en las Ecuaciones 2.9 y 2.13 se tiene que los pesos son

$$WH_{jk}^{nuevo} = WH_{jk}^{viejo} + \gamma(t_k - s_k)([1 - f(\sigma_k)]f(\sigma_k))h_j$$
$$WI_{ij}^{nuevo} = WI_{ij}^{viejo} + \gamma([1 - f(\sigma_j)]f(\sigma_j))I_i\Big[\sum_{l=1}^{K}(t_l - s_l)([1 - f(\sigma_l)]f(\sigma_l))WH_{jl}\Big] \quad (2.14)$$

Así obtenemos como es que se adaptarán numéricamente todos los pesos al usar la función sigmoide y es lo que se va a programar mas adelante. Si se utiliza la función tangente hiperbólica como función de activación se tiene que

$$f(x) = \frac{Exp[x] - Exp[-x]}{Exp[x] + Exp[-x]} \Rightarrow f'(x) = 1 - \frac{(Exp[x] - Exp[-x])^2}{(Exp[x] + Exp[-x])^2} = 1 - [f(x)]^2$$

por lo que los pesos para este caso se actualizarán de la siguiente manera

$$WH_{jk}^{nuevo} = WH_{jk}^{viejo} + \gamma(t_k - s_k)(1 - [f(\sigma_k)]^2)h_j$$
$$WI_{ij}^{nuevo} = WI_{ij}^{viejo} + \gamma(1 - [f(\sigma_j)]^2)I_i\Big[\sum_{l=1}^{K}(t_l - s_l)(1 - [f(\sigma_l)]^2)WH_{jl}\Big] \quad (2.15)$$

Una vez que se han actualizado los pesos, se propaga la información de entrada nuevamente y se usa este proceso hasta que el error se vuelva lo suficientemente pequenño como se quiera o sea posible, tal que los valores que se obtienen en las unidades de salida sean $s_k \approx t_k$ para toda k. A este proceso se le llama entrenamiento de la red, ya que una vez que termina el proceso, al introducir la información con la que se entrenó se obtendrán los valores deseados e incluso se puede extrapolar los resultados para conjuntos de datos que no se han entrenado pero que son similares a los que si fueron entrenados.

Los pasos a seguir entonces para usar una red tipo Backpropagation con una capa oculta es el siguiente

- Se introduce la lista o vector de datos que se quieren procesar, en el que cada elemento representará una unidad de entrada en la red (se toma la longitud de este vector para tener el número de neuronas en la capa de entrada en donde todos los vectores deberán tener la misma longitud), se selecciona el numero de neuronas en la capa oculta y en la de salida.

- Se seleccionan los pesos iniciales de la red aleatoriamente, en un intervalo por determinar.
- Se escoge una función de activación con la que red trabajará.
- Se selecciona el número de iteraciones que se realizarán para entrenar la red con el conjunto de datos introducidos inicialmente, el valor de la constante de aprendizaje, los bias, la constante del término de momento (se introducirá en el capítulo de resultados) y los valores que tendrán las neuronas objetivo.
- Se propaga la información obteniendo los valores de las neuronas h_j y s_k, utilizando las Ecuaciones 2.4 y 2.5 respectivamente.
- Ya que se propagó la información, se guarda el error (Ecuación 2.6) en un vector que contendrá el error para cada iteración.
- Se adaptan los pesos utilizando el par de Ecuaciones 2.14 o 2.15, dependiendo si se usó la función de activación sigmoide o tangente hiperbólica.
- Una vez que termina el entrenamiento de la red el número de iteraciones seleccionado, se grafica la función de error para observar su comportamiento.
- Finalmente si se observa que el error converge a cero, se guardan las matrices con los valores de los pesos y se introducen los datos que se entrenaron para ver los resultados obtenidos.

Si el error no converge a cero u oscila demasiado una vez que terminó el proceso, se cambia el conjunto de parámetros y se repite el proceso hasta obtener los resultados deseados.

Ilustraremos este proceso mediante un ejemplo sencillo, utilizando una red con dos neuronas en la capa de entrada, una neurona en la capa oculta y una neurona en la capa de salida. Los valores de los parámetros junto con los valores de las neuronas de entrada y objetivo se tomaron solo para ilustrar el proceso (no tienen ningún significado físico, ni representan algún tipo de variable) y se pueden ver en la Figura 2.9. Los pesos iniciales se seleccionaron aleatoriamente en el intervalo $[0, 1]$ (la razón de

tomarlos aleatoriamente se discutirá en el capítulo de los resultados). Como primer paso se propaga la información para encontrar los valores de las neuronas h_1 y s_1, utilizando la función sigmoide como función de activación. Los valores que resultan de este procedimiento son:

$$h_1 = f(\sum_{i=1}^{2} WI_{i1}I_i) = f(0.1 * 0.25 + 0.2 * 0.5) = f(0.125) = \frac{1}{1 + e^{-0.1257}} = 0.5312$$

$$s_1 = f(WH_{11}h_1) = f(0.05 * 0.53) = f(0.0265) = \frac{1}{1 + e^{-0.0265}} = 0.5066$$

Posteriormente se calcula el error y se almacena para utilizarlo mas adelante. Finalmente se actualizan los pesos con la Ecuación 2.14:

$$\begin{aligned} WH_{11}^{nuevo} &= WH_{11}^{viejo} + \gamma(t_1 - s_1)([1 - f(WH_{11} * h_1)]f(WH_{11} * h_1))h_1 \\ &= 0.05 + 10(0.4 - 0.5066)([1 - f(0.05 * 0.5312)]f(0.05 * 0.5312)) * 0.5312 \\ &= -0.0915 \end{aligned}$$

$$\begin{aligned} WI_{11}^{nuevo} &= WI_{11}^{viejo} + \gamma([1 - f(WI_{11} * I_1 + WI_{21} * I_2)]f(WI_{11} * *I_1 + WI_{21} * I_2)) * \\ &\quad I_1[(t_1 - s_1)(1 - f(WH_{11} * h_1))f(WH_{11} * WH_{11})] \\ &= 0.1 + 10([1 - f(0.1 * 0.25 + 0.2 * 0.5)]f(0.1 * 0.25 + 0.2 * 0.5) * \\ &\quad 0.25[(0.4 - 0.5066)(1 - f(-0.0915 * 0.5315))f(-0.0915 * 0.5315)(-0.0915) \\ &= 0.1015 \end{aligned}$$

$$\begin{aligned} WI_{21}^{nuevo} &= WI_{21}^{viejo} + \gamma([1 - f(WI_{11} * I_1 + WI_{21} * I_2)]f(WI_{11} * *I_2 + WI_{21} * I_2)) * \\ &\quad I_1[(t_1 - s_1)(1 - f(WH_{11} * h_1))f(WH_{11} * WH_{11})] \\ &= 0.2 + 10([1 - f(0.1 * 0.25 + 0.2 * 0.5)]f(0.1 * 0.25 + 0.2 * 0.5) * \\ &\quad 0.5[(0.4 - 0.5066)(1 - f(-0.0915 * 0.5315))f(-0.0915 * 0.5315)(-0.0915) \\ &= 0.2030 \end{aligned}$$

Ya que terminamos de adaptar los valores de las neuronas y los pesos se repite el procedimiento un total de 20 veces para obtener el resultado mostrado en la Figura 2.10 (izquierda). Se guardan los pesos y se utilizan para propagar un conjunto de datos diferentes a los que se entrenaron.

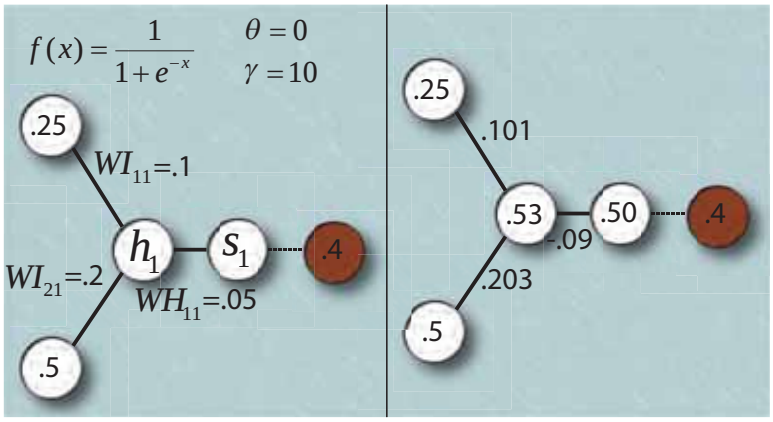

Figura 2.9: Ejemplo del proceso de entrenamiento de una RNA tipo backpropagation. A la izquierda se muestran los valores y parámetros con los que se inicializa la red, fijando los valores de las neuronas de entrada (0.25, 0.5) y objetivo (0.4) e inicializando aleatoriamente los pesos. A la derecha se muestran los valores de las neuronas h_1 y s_1 después de propagar una vez los datos de entrada, así como los pesos después de actualizarlos.

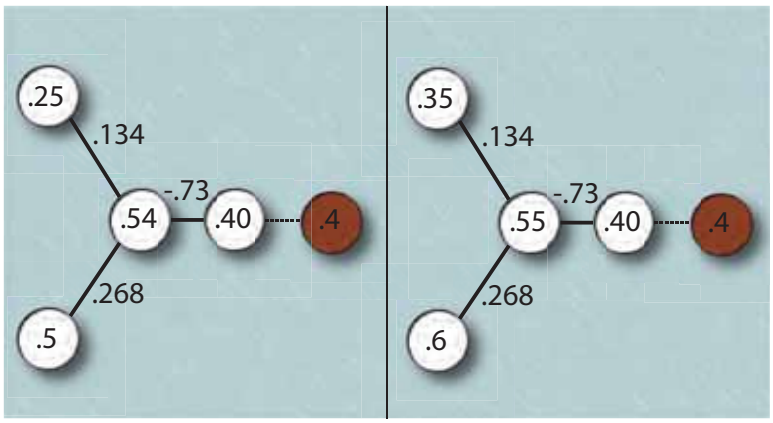

Figura 2.10: Resultados después de 20 iteraciones en la RNA tipo backpropagation. A la izquierda se muestran los valores que se obtienen en las neuronas h_1 y s_1 y los pesos después del entrenamiento. A la derecha se muestran los valores de las neuronas h_1 y s_1 al usar los pesos que resultaron del entrenamiento y utilizar otros datos de entrada.

Podemos observar en la Figura 2.10 (derecha), que aunque los datos de entrada son

diferentes a los que se usaron para el entrenamiento, se obtiene un valor cercano al objetivo propuesto en la neurona de salida (s_1=0.4004).

Así podemos apreciar con este ejemplo sencillo, la capacidad que tienen las RNA para extrapolar los resultados que se obtienen para conjuntos de datos distintos con los que se alimenta a la misma, a pesar de ser diferentes a los que se usaron para el entrenamiento de la red. Que tan buena es esta extrapolación dependerá de la cantidad de patrones diferentes que se usen para el entrenamiento y de que tanto difieran los datos que se introduzcan en la red con los que fueron entrenados.

Ya que sabemos como funciona el proceso de entrenamiento para una RNA tipo Backpropagation y los diferentes parámetros que intervienen en este proceso, pasaremos ahora a estudiar las señales digitales, particularmente las que representan información proveniente de una señal analógica de audio, así como también las transformadas de Fourier que nos servirán para hacer una selección de los datos que se introducirán en la red.

CAPÍTULO 3

EL SONIDO COMO UNA SEÑAL DIGITAL

En este capítulo se estudiará como es el proceso para convertir una onda sonora en una señal digital, para posteriormente poder procesar esta señal de tal manera que podamos encontrar información característica para introducirla en la RNA que desarrollemos, ya que como veremos, la información contenida en un segundo de audio, puede ser una lista con decenas de miles de entradas y la cual no podríamos introducirla directamente a la red debido al costo computacional que conlleva.

Para poder convertir el audio análogo a digital primero necesitamos tener una idea general de lo que es el sonido. El sonido es un fenómeno que necesita de un medio para propagarse, como puede ser el aire, el agua o un sólido y básicamente es una perturbación de este medio debido a cambios de presión en el mismo, el cual transporta energía pero no materia. Estas perturbaciones ocurren en forma de ondas longitudinales, es decir, la dirección en que se propagan es la misma que la dirección en la que ocurre la perturbación. Imaginemos que tenemos una fuente sonora, como puede ser una bocina Figura 3.1, que cuando genera algún sonido el cono que tiene se mueve hacia adelante y hacia atrás rápidamente, logrando que las partículas de aire frente al cono se compriman debido al cambio de presión y transmitan esta energía hacia las partículas vecinas que tienen frente a ellas, para posteriormente oscilar con respecto al punto donde estaban inicialmente. A su vez, las partículas vecinas que estaban enfrente

se comprimen y vuelven a generar el mismo proceso, así sucesivamente propagándose la onda por el espacio.

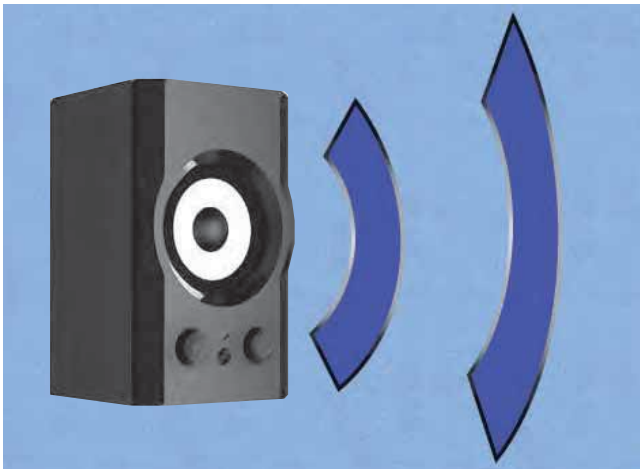

Figura 3.1: El movimiento del cono de la bocina produce cambios de presión en el aire que se propagan por el espacio.

Estas perturbaciones van cambiando en el tiempo y cualquier sonido en general se puede expresar como una suma de diferentes amplitudes y frecuencias para cada instante de la siguiente manera:

$$x(t) = \sum_{i=1}^{M} A_i(t) sen(2\pi\nu_i(t) + \theta_i(t))$$

donde $A_i(t)$, $\nu_i(t)$ y $\theta_i(t)$ son las amplitudes, frecuencias y fases (posiblemente variables en el tiempo) respectivamente de la onda sonora y M depende de la complejidad del sonido (si es música por ejemplo, esta M será muy grande). Cualquier cantidad física que dependa del tiempo, espacio o cualquier otra variable independiente es una señal y se describen por medio de funciones como la que acabamos de mostrar. Por esta razón el sonido es considerado una señal y más específicamente, una señal análoga debido a que las amplitudes y frecuencias son continuas. La generación de señales usualmente esta asociada a un sistema que responde a algún estímulo, en el ejemplo antes mencionado el sistema es la bocina, ya que responde a impulsos eléctricos para mover el cono y generar el sonido. Un sistema también puede ser considerado como un dispositivo que

realiza algún tipo de operación sobre la señal, como los filtros que eliminan ruido o interferencia en la transmisión de datos y los cuales modifican la señal que reciben. A este proceso se le llama procesado de una señal.
Para convertir una señal analógica a una digital se necesitan tres procesos:

- Muestreo. En este proceso se discretiza la señal en el tiempo, tomando ciertas muestras de la señal continua en determinados intervalos. Si $x(t)$ es la señal análoga de entrada al dispositivo o software que tomará las muestras, entonces la salida producida es $x(n)$, con $n = 0, 1, \ldots, N-1$, siendo N el número total de muestras. En este punto las muestras aún contienen amplitudes continuas.

- Discretización. Es el proceso que se realiza para pasar de la señal $x(n)$ con amplitudes continuas, a una señal $x_q(n)$ que solo puede tomar un valor específico de un conjunto finito de posibles valores. La diferencia entre la señal discretizada $x_q(t)$ y la señal continua en amplitud $x(n)$, se llama error de cuantización.

- Codificación. En el proceso de codificación, cada valor discreto $x_q(n)$ es representado por una secuencia de bits (1 y 0) para ser manipulado por la computadora.

Con estos tres pasos convertimos una señal analógica en una digital para poder utilizar un sistema de procesamiento mediante una computadora (más detalles [Orfanidis, 2010]). Este proceso lo realiza automáticamente el codificador en la computadora (la tarjeta de sonido) o en el dispositivo que se usa para realizar la grabación y nosotros no realizaremos esta tarea.
Siguiendo con el ejemplo que se pensó de la bocina, estas perturbaciones en el aire que se producen llegan a nosotros y las percibimos por medio de nuestros oídos. Los cambios de presión hacen que vibre una membrana flexible llamada tímpano y que posteriormente se convierten en señales eléctricas con ayuda de nuestras neuronas para que nuestro cerebro las interprete. Un procedimiento similar ocurre cuando se digitalizan estas senáles. Esta tarea la realizan los transductores, que son instrumentos que convierten energía mecánica en energía eléctrica. En el caso del sonido, los cambios de presión (o de amplitudes) son transformados en señales eléctricas utilizando un micrófono que hacen el papel del tímpano y registran tales cambios en intervalos discretos de tiempo mediante un diafragma. Este diafragma esta conectado a un dis-

positivo dependiendo del tipo de micrófonno, por ejemplo a una bobina, que se encarga de convertir los registros sonoros obtenidos en señales eléctricas.

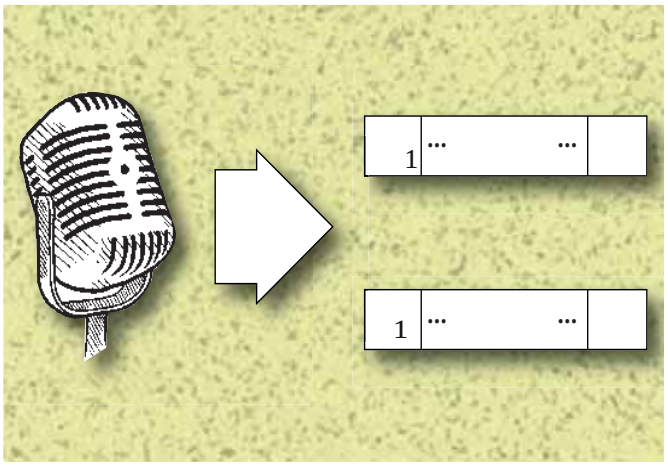

Figura 3.2: El micrófono transforma la energía mecánica de las ondas sonoras en impulsos eléctricos, captando las amplitudes en intervalos discretos de tiempo al digitalizar la señal analógica.

Después que tenemos digitalizada la señal, tendremos una lista de valores con las amplitudes para cada instante $t = \frac{nT}{N}$, con $0 \leq t \leq T$, siendo T el tiempo total de duración de la señal (Figura 3.2). Pero, ¿Cómo sabemos que tantas muestras tomar en el intervalo de tiempo que dura la señal? Esta respuesta nos la da el teorema de Nyquist, el cual establece que para que una señal $x(t)$ pueda ser representada adecuadamente por $x(n)$ se deben cumplir dos cosas:

1. La señal $x(t)$ debe estar limitada en el espectro de frecuencias, es decir, que exista una frecuencia máxima ν_{max} tal que la señal no tenga frecuencias mayores a esta.

2. La tasa de muestreo $\nu_s = \frac{1}{N}$ debe ser al menos el doble de la frecuencia máxima: $\nu_s \geq 2\nu_{max}$.

El valor mínimo de la tasa de muestreo $\nu_s = 2\nu_{max}$ se llama tasa de Nyquist. Al intervalo $[-\frac{\nu_s}{2}, \frac{\nu_s}{2}]$ se le conoce como intervalo de Nyquist.

Una vez que se fija la frecuencia de muestreo ν_s podemos relacionarla con los intervalos de tiempo en donde se toman las muestras :

$$t_n = \frac{n}{N}T = \frac{n}{N}(\frac{N}{\nu_s}) = \frac{n}{\nu_s} \tag{3.1}$$

Para fijar una frecuencia de muestreo apropiada para cuando queremos digitalizar una señal de sonido, tenemos que tener en cuenta el rango de frecuencias que los humanos alcanzamos a percibir con nuestros oídos, el cual en promedio va desde los 20 Hz hasta los 20 KHz. Por esta razón la frecuencia de muestreo teórica mínima que debemos de tomar, es de $\nu_s = 2\nu_{max} = 40$ KHz. En general se suele utilizar una frecuencia de muestreo de 44100 Hz, esto con fines prácticos, ya que usualmente los codificadores usan filtros que agilizan los cálculos usando este rango de frecuencias adicionales.

Una vez que tenemos la señal digitalizada ya podemos utilizar herramientas computacionales para su análisis como es el caso de la transformada de Fourier o la transformada z. Se optó por usar la transformada de fourier, por que además de facilitar el análisis de audio también esta íntimamente ligado a las señales digitales [Hayes, 1999] y nos serán de utilidad al querer estudiar otro tipo de señales. La transformada de Fourier nos sirve para representar cualquier señal en términos de señales sinusoidales (o exponenciales complejas). Para ilustrar este proceso, podemos pensar en cuando un rayo de luz atraviesa un prisma y se descompone en los diferentes colores visibles, o lo que es lo mismo, una superposición de ondas de diferentes frecuencias. Un proceso similar ocurre cuando utilizamos una transformada de Fourier descomponiendo la señal como una suma de señales de distintas frecuencias.

3.1 Transformada de Fourier

La transformada de Fourier surge al pensar en la expansión de una función $f(t)$ contínua, periódica (de periodo T) y de cuadrado integrable, en términos de una suma de funciones exponenciales complejas

$$f(t) = \sum_{n=-\infty}^{\infty} C_n Exp[i\omega_0 nt] \tag{3.2}$$

donde ω_0 es la frecuencia $\omega_0 = \frac{2\pi}{T}$ y C_n son los coeficientes de Fourier que están dados por

$$C_n = \frac{1}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} f(t)Exp[-i\omega_0 nt]dt \tag{3.3}$$

y se encuentran usando la teoría de funciones ortogonales [Proakis and Manolakis, 2006]. A la serie infinita de la Ecuación 3.2 se conoce como serie de Fourier. Este es un caso muy particular ya que no siempre se tienen funciones periódicas, pero se puede generalizar a cualquier tipo de funciones (siempre y cuando sean de cuadrado integrable) y pensar en la función $f(t)$ en donde esta definida, como una función que tiene un periodo infinito. Entonces cuando $T \to \infty$, el término $\omega_0 n \to \omega$, es decir, pasamos a un espectro continuo de frecuencias y la integral en la Ecuación 3.3 se convierte en

$$F(\omega) = \int_{-\infty}^{\infty} f(t)Exp[-i\omega t]dt \tag{3.4}$$

A la ecuación anterior se le llama transformada de Fourier de la función $f(t)$. Así pasamos de tener una función que depende de la variable t a una función que depende de la variable ω. Por esta razón se dice comúnmente que pasamos al espectro de frecuencias. La función $F(\omega)$ tiene la característica de que es una función periódica con periodo igual a 2π. Por tal motivo se suele analizar únicamente el intervalo $0 \leq \omega \leq 2\pi$, nombrado el intervalo principal.

Hasta aquí hemos usado el hecho de que se tiene una función continua $f(t)$, pero nosotros tenemos una función discreta $f(t) = \{f(t_0), f(t_1), \ldots, f(N-1)\}$ por lo que necesitaremos de la transformada de Fourier para el caso de una función discreta.

3.2 Transformada discreta de Fourier

Se ha usado hasta este momento una función arbitraria $f(t)$. De ahora en adelante se usará la notación para representar la función discreta (muestras de audio tomadas) con: {x(0),. . . ,x(N-1)} y la transformada de Fourier $F(\omega)$ la asociaremos con N valores tal que $X(\omega) = \{X(0), \ldots, X(N-1)\}$ es la transformada de Fourier de x(n).
Queremos pasar entonces al espectro de frecuencias, por lo que se desea tener N valores de la función $X(\omega)$. Como esta función tiene periodo 2π, se requiere dividir este intervalo en N partes iguales y haciendo $\omega = \frac{2\pi k}{N}$ con $k = 0, 1, \ldots, N-1$ se logra este

objetivo. Si pensamos en nuestra señal inicial como una serie infinita de valores tales que

$$\text{x(n)} = \begin{cases} \text{x(n)} & si \quad 0 \leq n \leq N-1, \\ 0 & en \quad otro \quad caso \end{cases}$$

podemos aproximar la integral en la transformada de Fourier (Ecuación 3.4) por una suma:

$$X(k) = \sum_{n=-\infty}^{\infty} x(n) Exp[\frac{-2\pi ikn}{N}] = \sum_{n=0}^{N-1} x(n) Exp[\frac{-2\pi ikn}{N}] \tag{3.5}$$

en donde k esta relacionada con los valores discretos que toman las frecuencias (de forma similar a como n representa los valores discretos en el tiempo) de la siguiente manera:

$$\nu = \frac{k}{N}\nu_s \tag{3.6}$$

Estos valores discretos de las frecuencias nos serán de utilidad mas adelante para hacer una selección de las mismas y que servirán como datos de entrada para la RNA que utilizaremos. De esta manera se obtiene la transformada discreta de Fourier (abreviado TDF) $X(k)$ para cada uno de los valores de la lista $x(n)$. Recordemos que N esta relacionado con la tasa de Nyquist y hay que tener esto en consideración para no obtener efectos de aliasing [Orfanidis, 2010].

Con la Ecuación 3.5 ya podríamos tratar de hacer los cálculos para encontrar la transformada de Fourier de los datos que contienen las muestras de audio, pero al necesitar un número muy grande de muestras por segundo, empleando esta ecuaión se necesitaría mucho tiempo de procesamiento. Es por esto que es necesario hacer uso de otras técnicas para disminuir el tiempo de cálculo de la TDF.

3.3 Transformada rápida de Fourier.

Para realizar la evaluación de una transformada discreta de Fourier maś eficiente, se pueden utilizar varios métodos. Varios de estos métodos usan la idea de "divide y vencerás" para separar la señal $x(n)$ a la que se le aplicará la TDF, en conjuntos de menor tamaño. La transformada rápida de Fourier consiste entonces en dividir la señal

$x(n)$ inicial en subconjuntos mas pequeños en donde se evaluará la transformada. Si expresamos la Ecuación 3.5 como

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk} \tag{3.7}$$

$$W_N = Exp[\frac{-2\pi i}{N}]$$

podemos ver que se necesitan realizar N multiplicaciones y N sumas para calcular la TDF $X(k)$, por lo que las operaciones a realizar son del orden de N. Como $0 \leq k \leq N-1$, se necesita evaluar un total de N veces las operaciones anteriores, por lo que el total de cálculos es del orden de N^2. Si dividimos las muestras en 2 grupos del mismo tamaño, es decir, dos conjuntos de longitud $\frac{N}{2}$ (si el número de muestras es impar se completa agregando un término igual a cero al final de la lista) y ahora calculamos las TDF para cada conjunto, con lo que se obtiene que cada transformada necesitará $(\frac{N}{2})^2$ operaciones. Entonces el total de operaciones a realizar son del orden de $\frac{1}{2}N^2$. Así vemos que podemos reducir el número de operaciones a la mitad. Este tipo de idea se usa en algoritmos para ordenamiento de listas por mezcla (merge sort en inglés) y como hemos visto, al dividir en subconjuntos mas pequeños la lista inicial, se puede reducir el número de operaciones que se tendría que hacer usando directamente la lista.
A esta manera de evaluar las TDF en conjuntos mas pequeños se les conoce con el nombre de transformada rápida de Fourier (FFT en ingés) y existen varios métodos dependiendo de como se dividan los subconjuntos.

Como nosotros estamos interesados en el funcionamiento de la RNA y como existen varias formas de llevar a cabo la FFT, se utilizó la librería incluida en Mathematica para la transformada de Fourier. Solo se ilustra el proceso de manera general en el que se transformarán los datos obtenidos de la grabación, para llevarlos al espacio de frecuencias en los que será mas fácil seleccionar información que nos sirva para caracterizar las palabras.

Continuaremos ahora con la implementación de la RNA, resolviendo un problema sencillo para comprobar el funcionamiento del algoritmo y posteriormente usar la red para procesar los datos obtenidos de las muestras de audio.

CAPÍTULO 4

RESULTADOS

Antes de pasar al objetivo central de este proyecto, que es el análisis de una señal digital de audio para el reconocimiento de voz, primero necesitamos implementar la RNA y realizar pruebas para comprobar que funciona correctamente. El software que se utilizó para desarrollar de la RNA fue Wolfram Mathematica 9, ya que tiene herramientas para manipulación de audio y se está familiarizado con el mismo, pero la migración a otro software si se quisiera hacer no es tan complicada, debido a que la red se construyó desde cero.

4.1 Implementación de la RNA Backpropagation

Lo primero que se hizo fue implementar la red neuronal en el software en el que se trabajó y probar que funciona con un ejemplo sencillo, pero también es en el que más se manipularon los parámetros para entender como modifican la red y aplicar estos conocimientos en los problemas con mayor complejidad analizados después.

Recordando lo que se estudió en el capítulo 2, en la capa de entrada de la RNA introduciremos datos que contengan información relevante del problema que se esta analizando. Para esto, tal vez sea necesario primero realizar algún tipo de transformación en los datos. Los problemas que presentaremos aquí, son problemas en los que queremos clasificar o identificar los diferentes conjuntos de datos o patrones introducidos en las neuronas de entrada, lo cual se realiza mediante la propuesta de los objetivos.

En otras palabras, sabemos lo que queremos obtener como resultado de propagar los datos desde las neuronas de entrada hacia las de salida, para cada conjunto de datos con los que se entrena a la red. En caso de no tener idea de los valores que se pueden obtener en las neuronas de salida, se debe recurrir a usar otro tipo de estructura de RNA.

Nosotros usaremos una idea similar a la que se usa en las redes perceptrón para análisis de caracteres para clasificar la información en este primer ejemplo. Lo que utilizan en ese caso es que cuando quieren reconocer alguna letra, utilizan los pixeles que la representan en pantalla, asignando un 0 a pixeles que no están encendidos y un 1 para los pixeles que si lo están, por lo que cualquier caracter es representado por una matriz de unos y ceros [Freeman and Skapura, 1991]. Como primer ejemplo se quiso representar de alguna forma números entre uno y diez, de tal manera que la red nos indique que número es el que se introdujo (aunque esto se puede realizar con una simple comparación sin la necesidad de codificar los números o de usar una RNA, lo que se hizo fue verificar el buen funcionamiento de la red). Con la idea anterior en mente se presentaron tanto los objetivos deseados como los números introducidos como una lista de unos y ceros, representando por ejemplo al número uno como una lista de diez entradas, en la que la primer entrada es uno y las demás cero. Si el número es el dos, la lista que lo representa es un uno en la segunda entrada y cero en las demás, así sucesivamente con todos los números hasta el diez. Para ejemplificar, tomemos la forma en que representamos el dos al introducirlo en las neuronas de entrada de la red:

$$2$$

$$\Downarrow$$

$$\{\ 0\ ,\ 1\ ,\ 0\ ,\ 0\ ,\ 0\ ,\ 0\ ,\ 0\ ,\ 0\ ,\ 0\ ,\ 0\ \}$$

De igual manera se propone que cuando se introduzca cada número con esta representación, en las neuronas de salida nos de como resultado el mismo vector, es decir, queremos identificar el número que se introdujo mediante el uso de los objetivos. De este modo si se introduce el 2 como anteriormente vimos, queremos que las neuronas de salida tengan valores de 0, excepto en la segunda neurona de salida, donde tendrá el valor de 1. Entonces si al propagar la información de los datos que representan al 2, en las neuronas de salida obtenemos un 1 en la segunda entrada y 0 en las demás,

1	objetivos	0.09	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
2	objetivos	0.01	0.09	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
3	objetivos	0.01	0.01	0.09	0.01	0.01	0.01	0.01	0.01	0.01	0.01
4	objetivos	0.01	0.01	0.01	0.09	0.01	0.01	0.01	0.01	0.01	0.01
5	objetivos	0.01	0.01	0.01	0.01	0.09	0.01	0.01	0.01	0.01	0.01
6	objetivos	0.01	0.01	0.01	0.01	0.01	0.09	0.01	0.01	0.01	0.01
7	objetivos	0.01	0.01	0.01	0.01	0.01	0.01	0.09	0.01	0.01	0.01
8	objetivos	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.09	0.01	0.01
9	objetivos	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.09	0.01
10	objetivos	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.09

Tabla 4.1: Valor de los objetivos propuestos, los cuales nos indican los valores que queremos que tengan las neuronas de salida una vez que la red se entrene y los pesos se hayan adaptado.

podremos identificar que número es el que se introdujo en la red.

Sólo que hay un inconveniente al usar esta idea: las funciones de activación que se usan en las redes tipo Backpropagation deben ser diferenciables y ademas su derivada no debe de ser constante, por lo que no podemos usar la función escalón como se haría en las redes perceptrón. Es por eso que es muy común usar las función sigmoide o la tangente hiperbólica, pero estas funciones se aproximan asintóticamente a 1 y nunca lo alcanzan, por lo que no podemos poner como objetivo el 1 o en el caso de la tangente hiperbólica el cero, ya que el valor de la función en este punto también es cero y los pesos no los podríamos actualizar a menos que se pusiera el término de bias. Por esta razón en vez de unos y ceros podemos pensar en cantidades como .01 o .1 en vez de cero y .09 o .9 en vez de uno para los valores propuestos en los objetivos o cualquier combinación que queramos que esté en el rango $[-1, 1]$ para el caso de la tangente hiperbólica o $[0, 1]$ en el caso de la función sigmoide, de tal manera que nos sirva para clasificar la información que se introduce en la red. Considerando estos echos, tomamos tanto los objetivos como la representación de los números de la manera que se muestra en la Tabla 4.1. Decidimos poner en las unidades de entrada los mismos valores que

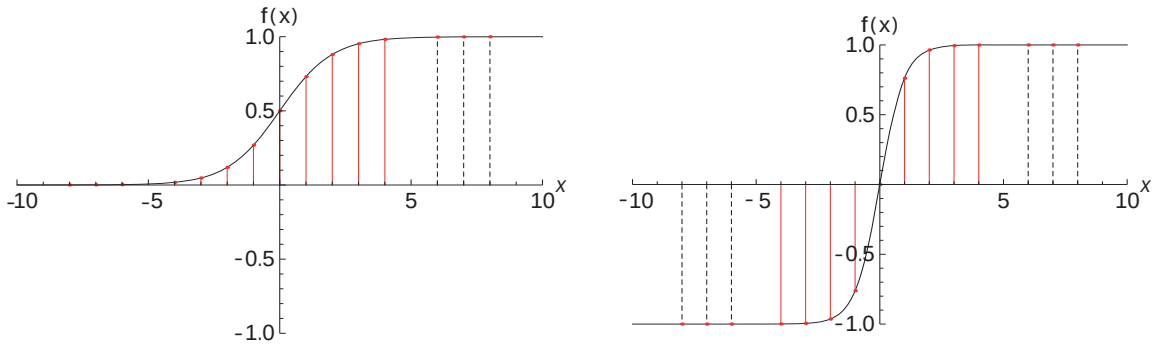

Figura 4.1: A la izquierda la función tangente hiperbólica y a la derecha la función sigmoide, donde las lineas continuas indican los puntos donde el valor de la función tiene mayor cambio y las lineas punteadas indican valores en donde la función tiene valores similares.

pusimos en los objetivos por comodidad, pero pudimos haber puesto cualquier cantidad dentro del intervalo donde las funciones de activación son mas sensibles (Figura 4.1). Nos dimos cuenta que es muy importante los valores que se introducen en las unidades de entrada y que si no están dentro de cierto rango, es necesario normalizar estas cantidades, ya que si no se normalizan puede ser que la red no funcione aunque modifiquemos parámetros como el número de neuronas, el parámetro de aprendizaje, los bias o el término de momento si es que se agregó. Es por esto que hay que tener en cuenta donde es mas susceptible la función de activación que se está usando, por ejemplo en el caso de la tangente hiperbólica, el intervalo en donde es más sensible es entre $[-2, 2]$, mientras que la función sigmoide el intervalo es entre $[-4, 4]$. Una vez que sabemos como vamos a introducir los datos de entrada y los objetivos se procedió a entrenar a la red cambiando los parámetros y viendo su comportamiento. Al inicio lo que se hizo fue entrenar en orden los números, primero el uno cien iteraciones, luego el dos cien veces, así hasta el diez, sumando en total mil iteraciones. De lo que nos percatamos fue de que una vez entrenada la red al introducir los conjuntos de datos que representan los números, los resultados eran erróneos y el único numero que se aproximaba a los objetivos fue el diez (Tabla 4.2).

1	2	3	4	5	6	7	8	9	10
0.0101	0.0102	0.0103	0.0105	0.0104	0.0106	0.0106	0.0104	0.0106	0.0102
0.0102	0.0104	0.0106	0.0105	0.0106	0.0106	0.0106	0.0106	0.0107	0.0104
0.0105	0.0105	0.0108	0.0106	0.0107	0.0109	0.0108	0.0107	0.0109	0.0104
0.0107	0.0110	0.0109	0.0110	0.0109	0.0111	0.0110	0.0110	0.0111	0.0107
0.0110	0.0112	0.0111	0.0115	0.0113	0.0114	0.0114	0.0113	0.0114	0.0111
0.0117	0.0118	0.0118	0.0117	0.0118	0.0121	0.0119	0.0120	0.0119	0.0115
0.0122	0.0125	0.0124	0.0128	0.0126	0.0126	0.0127	0.0126	0.0128	0.0124
0.0139	0.0141	0.0142	0.0142	0.0142	0.0142	0.0143	0.0143	0.0144	0.0139
0.0179	0.0181	0.0182	0.0179	0.0181	0.0185	0.0183	0.0183	0.0185	0.0177
0.0899	0.0899	0.0911	0.0907	0.0920	0.0916	0.0909	0.0917	0.0913	0.0900

Tabla 4.2: Valores de las neuronas de salida al introducir la representación para cada número después de entrenarlos en orden. Los parámetros usados son: función de activación $f(x) = \frac{1}{1+Exp[-x]}$, pesos iniciales aleatorios entre $[0, 1]$, neuronas en la capa oculta=10, $\gamma = 10$, $\theta = 0$ y $\mu = 0$. El único número que da como resultado los objetivos propuestos para reconocerlo es el 10, debido a que fue el último que se entrenó.

Esto nos indica que el orden en que se introducen los conjuntos de datos para entrenar la red también importa y la red se adaptó mejor al número que se entrenó al final, por lo que posteriormente se entrenaron los números de forma aleatoria. De aquí en adelante los conjuntos de datos que representan las palabras, se introducirán en orden aleatorio dentro de la red para el entrenamiento.

Posteriormente se prosiguió a modificar los diferentes parámetros: el intervalo en donde se inicializan los pesos aleatorios, el número de neuronas en la capa oculta, el parámetro de aprendizaje, las funciones de activación y los bias. En este ejemplo no se introdujo término de momento, por que se quiere iniciar con lo más básico e ir complicando el problema poco a poco.
En la Figura 4.2 podemos observar como varía el error, al entrenar la red con cien iteraciones y cambiar el parámetro de aprendizaje dejando los demás parámetros constantes.

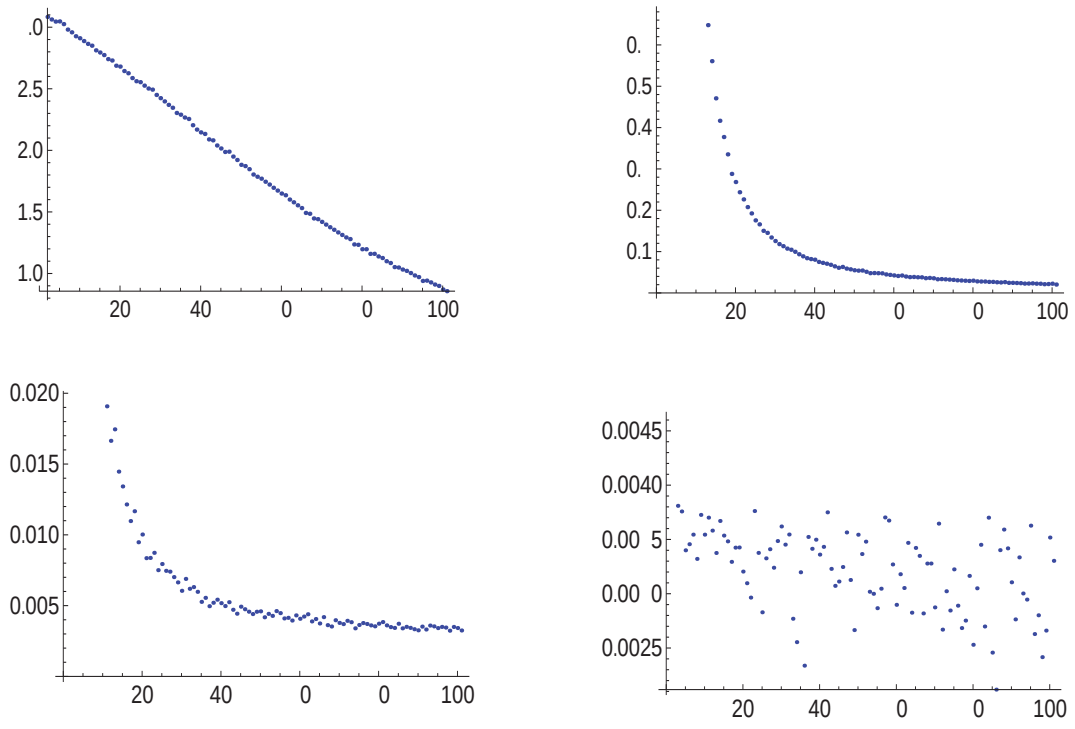

Figura 4.2: Comportamiento del error al entrenar 100 veces la red, utilizando la función sigmoide, 5 neuronas en la capa oculta, $\theta = 0$, $\mu = 0$ y variando únicamente el parámetro de aprendizaje. Arriba a la izquierda se utilizó $\gamma = 0.1$. Arriba a la derecha $\gamma = 1$. Abajo a la izquierda $\gamma = 10$. Abajo a la derecha $\gamma = 100$. Lo que nos muestra que entre mayor es γ, mayor la oscilación del error.

Si el error se aproxima a cero lentamente, al aumentar γ vemos como el error tiende más rápido a cero, hasta llegar un momento en donde al utilizar valores muy grandes, el error empieza a oscilar. Preferimos que el error oscile menos debido a que, cuando el error oscila, en ocasiones se queda oscilando y ya no converge a cero o lo hace de manera más lenta (aunque se podría usar un comportamiento que oscila una vez que se alcanzó un error mínimo). Se utilizó después el valor de $\gamma = 1$, con el que oscila menos el error, realizando tres mil iteraciones (no tenía caso utilizar un número mayor de iteraciones ya que llegó un momento donde el error ya no disminuyó) para el entrenamiento de la red y

se usaron los pesos que resultaron del entrenamiento para ver como respondían los datos al introducirlos, pero no dieron los objetivos esperados. Esto es posiblemente a que el número de neuronas en la capa oculta no es suficiente y se necesita un número mayor para que existan las suficientes conexiones, haciendo posible que se puedan alcanzar los objetivos propuestos. Por este motivo se incrementó el número de neuronas hasta un total de veinte tomando los parámetros anteriores ($f(x) = \frac{1}{1+Exp[-x]}$, pesos entre $[0, 1]$, $\gamma = 1$, $\theta = 0$) para probar la red depués de entrenarla con 3000 iteraciones, pero no funcionó al momento de reconocer los números. Lo que se observó en este proceso es que al incrementar el número de neuronas dejando los demás términos constantes, el error disminuyó pero no muy considerablemente (pasó de tener un error de aproximadamente 0.0029 a 0.0028).

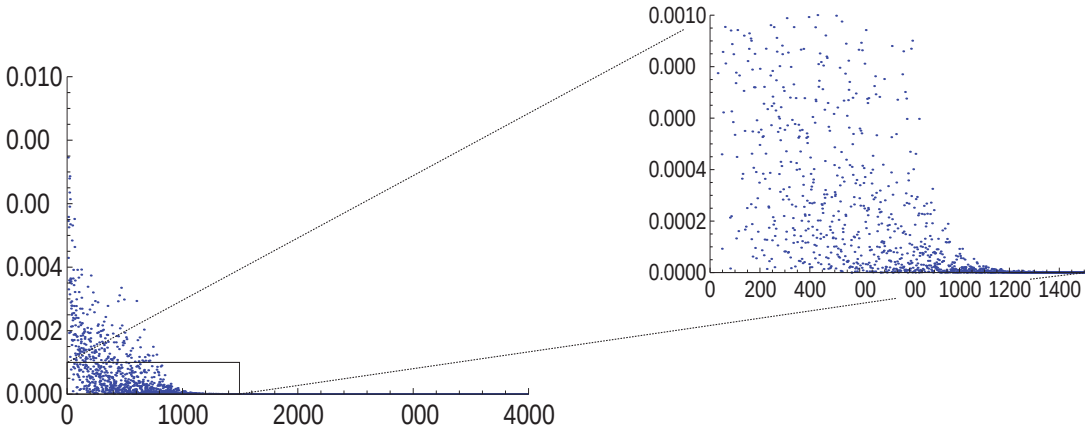

Figura 4.3: Error $E = \frac{1}{2}\sum_{k=1}^{salida}(t_k - s_k)^2$ de los 10 números para 4000 iteraciones. Función de activación: $\frac{Exp[x]-Exp[-x]}{Exp[x]+Exp[-x]}$, neuronas/capa oculta:10, objetivos: {.09,.01}, $\theta : 0$, $\gamma : 10$, pesos iniciales: [0,1]. Podemos ver que el error converge a cero.

Una vez que vimos que no hubo mejoras al incrementar el número de neuronas en la capa oculta, se cambió la función de activación por la tangente hiperbólica, comenzando las pruebas nuevamente con cinco neuronas en la capa intermedia y usando $\gamma = 0.1$ que fue el parámetro que se encontró era el óptimo (con $\theta = 0$ y pesos entre $[0, 1]$). Se realizaron diez mil iteraciones ya que el error variaba muy lentamente, pero no se

1	2	3	4	5	6	7	8	9	10
0.0901	0.0099	0.0099	0.0104	0.0101	0.0101	0.0100	0.0101	0.0100	0.0102
0.0099	0.0899	0.0099	0.0100	0.0099	0.0100	0.0099	0.0100	0.0100	0.0099
0.0100	0.0099	0.0899	0.0101	0.0100	0.0100	0.0099	0.0100	0.0100	0.0100
0.0100	0.0099	0.0099	0.0902	0.0100	0.0101	0.0099	0.0100	0.0100	0.0101
0.0102	0.0099	0.0099	0.0106	0.0901	0.0101	0.0100	0.0101	0.0100	0.0103
0.0104	0.0099	0.0099	0.0108	0.0102	0.0902	0.0100	0.0102	0.0100	0.0105
0.0104	0.0099	0.0099	0.0109	0.0102	0.0102	0.0900	0.0102	0.0100	0.0105
0.0103	0.0099	0.0099	0.0108	0.0102	0.0102	0.0100	0.0902	0.0100	0.0105
0.0102	0.0099	0.0099	0.0105	0.0101	0.0101	0.0100	0.0101	0.0900	0.0103
0.0103	0.0099	0.0099	0.0107	0.0102	0.0102	0.0100	0.0101	0.0100	0.0904

Tabla 4.3: Valores de las neuronas de salida al introducir la representación para cada número después de entrenarlos 1500 veces. Función de activación $\frac{Exp[x]-Exp[-x]}{Exp[x]+Exp[-x]}$, neuronas/capa oculta:10, objetivos: {.09,.01}, $\theta : 0$, $\gamma : 10$, pesos iniciales: [0,1]. El número de neurona que se "activa" (contando de arriba hacía abajo) con valor cercano a 0.09, corresponde al número que se introdujo (la lista de 10 valores que representa tal número).

logró un buen funcionamiento de la red (le faltó detectar solo dos números de los diez). Recurriendo a la experiencia adquirida en las pruebas anteriores, se aumentó el numero de neuronas para que el error se aproximara mas rápidamente a cero. Con un total de diez neuronas en la capa oculta y ajustando el parámetro de aprendizaje a $\gamma = 10$ para el entrenamiento de la red con 4000 iteraciones, se obtuvo el desempeño óptimo de la red para este problema, dando como resultado exactamente los objetivos que se seleccionaron (Figura 4.3).

Al realizar 1500 iteraciones ya funcionaba correctamente la red aunque con valores aproximados de salida (Tabla 4.3). Para fines prácticos ya se podría considerar como un buen resultado, ya que este tipo de resultados es más común en problemas en donde los datos de entrada son mas variados (en este ejemplo de entrada solo tenemos valores de 0.09 y 0.01) como los que veremos más adelante. Teniendo los parámetros con los que la red funciona de manera óptima, se analizó el comportamiento de la misma, ahora

al cambiar los valores con los que se inicializan los pesos y los bias para así saber si se mejoran los resultados o empeoran. Lo que sucede al variar el intervalo donde se generan aleatoriamente los pesos, es que depende de la combinación de los datos de entrada con los pesos y de los objetivos propuestos. Nos referimos a que por ejemplo, si pusimos como objetivos .09 y .01 entonces el argumento de la función de activación debe ser tal que $f(x) = .09$ o $f(x) = .01$ (o quedar cerca para que el proceso de adaptar los pesos se aproxime mas rápido a los valores objetivo), por lo que $x = \sigma_k = \sum_j H_j W H_{jk}$, en donde a su vez cada H_j dependen de la combinación de pesos con los datos de entrada ($H_j = f(\sigma_j) = f(\sum_i I_i W I_{ij})$). En la práctica si depende en gran medida del valor inicial de los pesos, ya que pueden mejorar los resultados o pueden incluso hacer que el error aumente, aún cuando se usen parámetros con los que ya se había resuelto el problema (Figura 4.4).

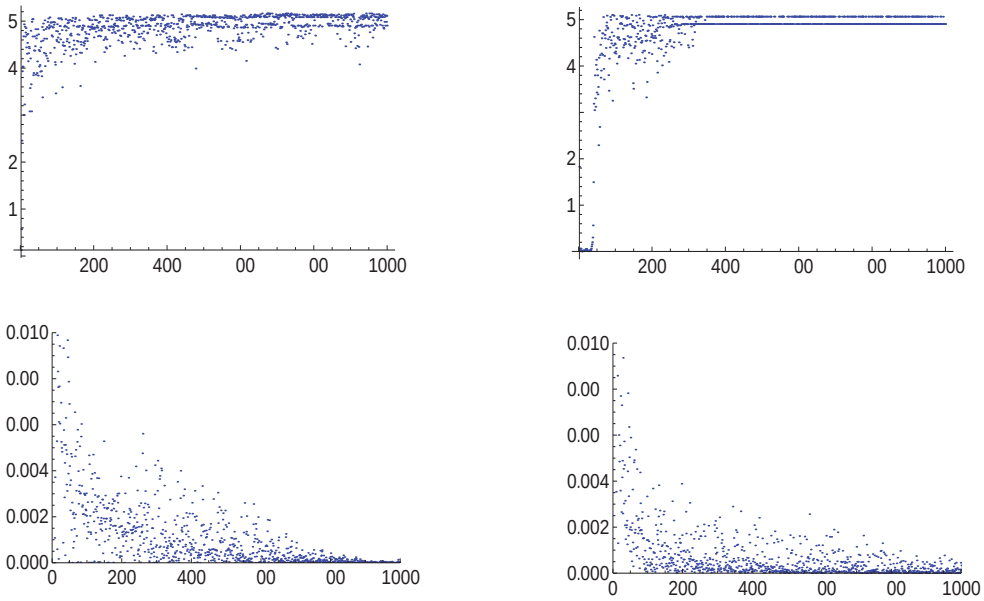

Figura 4.4: Arriba a la izquierda el error no converge a cero al poner pesos iniciales constantes ($WI = 1, WH = .1$). Arriba a la derecha el error no converge a cero con pesos aleatorios iniciales entre $[0.2, 1]$. Abajo a la izquierda el error converge a cero con pesos iniciales aleatorios ($WI = [-1, 0], WH = [0, 1]$). Abajo a la derecha el error converge a cero para pesos iniciales aleatorios en $[-1, 1]$.

Podemos ver que al poner pesos iniciales con valor constante el error no converge a cero. También aunque pongamos pesos iniciales aleatorios pero si no es un intervalo adecuado diverge. Puede incluso converger el error mas rápido a cero si se ponen pesos iniciales negativos en una matriz y positivos en la otra. Con estas pruebas podemos saber en el futuro que si el error no converge y es muy grande, la causa puede ser debido a una mala elección de los pesos iniciales que se tomaron.

Por último se vió el comportamiento de la red al cambiar los valores de los bias. Se observó que si asignamos valores iguales para los bias, ya sea en las unidades sigma que se usan para obtener los valores de las neuronas en la capa oculta ($\sigma_j = \sum_i W_{ij} I_i + \theta_j$), o en las unidades sigma que se usan para las neuronas en la capa de salida ($\sigma_k = \sum_j W_{jk} H_j + \theta_k$) en general desestabilizan la red, aún usando valores pequeños como $\theta_j = .1 = \theta_k$ o $\theta_j = .01 = \theta_k$. Poniendo todos los valores de los bias en cero, excepto alguno de los θ_j, también lleva a la divergencia del error, esto debido a que al modificar alguna neurona en la capa oculta con el término constante, éste afecta a todas las neuronas en la capa de salida. Si usamos todos los bias igual a cero, salvo alguno de los θ_k, en ocasiones la red se conserva funcionando correctamente. Es por esto que estos términos constantes se tienen que manejar con mucho cuidado y utilizarlos solo en problemas específicos donde se tenga idea donde pueden ir.

Ya hemos analizado el comportamiento de casi todos los parámetros involucrados en la red (falta ver cuando ponemos el término de momento) y con el conocimiento adquirido podremos tener mejor criterio al modificar la red para los problemas más complicados que veremos a continuación.

4.2 RNA para reconocimiento de voz usando amplitudes

Pasaremos ahora a analizar el problema principal de este proyecto: dada un grabación que contiene la voz de alguna persona de entre un grupo de estas, identificar a quién pertenece tal voz. La idea es seleccionar a un grupo de personas para grabar ciertas palabras que se les pedirá que digan. Una vez teniendo estas grabaciones, se les tendrá que realizar un proceso para obtener información característica de cada palabra, que posteriormente será utilizada como datos de entrada para la RNA. Se tendrán que seleccionar objetivos para poder clasificar la información introducida y saber a que persona pertenece la voz. Para poder utilizar la red neuronal en el reconocimiento de voz, necesitamos una forma de caracterizar la información que tenemos de tal manera que podamos introducirla en la red, ya que como se mencionó anteriormente la normalización de los datos juega un papel fundamental. Para las grabaciones se utilizó una frecuencia de muestreo de $\nu_s = 44100$ Hz, para abarcar todo el espectro auditivo, pensando a futuro para poder analizar cualquier tipo de sonidos y no sólo voces. De aquí en adelante al hablar de las frecuencias, nos referimos a las frecuencias discretizadas de la Ecuación 3.6 y que estarán relacionadas con la frecuencia de muestreo de la siguiente forma:

$$\nu = \nu_k = \frac{k}{N}\nu_s = \frac{k}{N}\Big(2.2\nu_{max}\Big) \tag{4.1}$$

donde $\nu_{max} = 20$ KHz, la maxima frecuencia audible. Así al decir que tenemos una frecuencia de 3000, queremos indicar el valor que tendrá al usar $k = 3000$.

Se requirió en un principio la ayuda de tres personas para grabar sus voces, las cuales se guardaron en formato wav, que es el tipo de archivos soportados por Mathematica y se utilizó un micrófono en donde las palabras no sufren de gran distorsión. De estas tres personas dos eran del sexo masculino (Personas 1 y 2) y una del sexo femenino (Persona 3) y se guardaron las voces para cada una de ellas con las palabras: hola, bienvenido, adiós, felicidades y su nombre. Más adelante se utilizará la voz de 3 personas más, pero se quiso ver si tendríamos que cambiar demasiado los parámetros usados con un menor número de personas y como verémos, si es necesario realizar varios cambios para lograr el funcionamiento adecuado de la red al utilizar mayor cantidad de datos para

el entrenamiento. Una vez teniendo los archivos de voz, se aplicó la FFT a la lista de datos que contiene la palabra hola dicha por cada persona. Esto se hace para facilitar el análisis de la información y para poder reducir la misma, después de obtener solo ciertos datos.

La idea para procesar la información es similar a la que se utilizó en el primer problema de los números: usar diez neuronas de entrada, una capa oculta con un numero de neuronas por determinar (se inciará con diez que fueron las necesarias en el problema anterior), tres neuronas en la capa de salida que nos ayudarán a clasificar los datos y objetivos con valores que nos ayuden a indicar a que persona pertenece la voz.

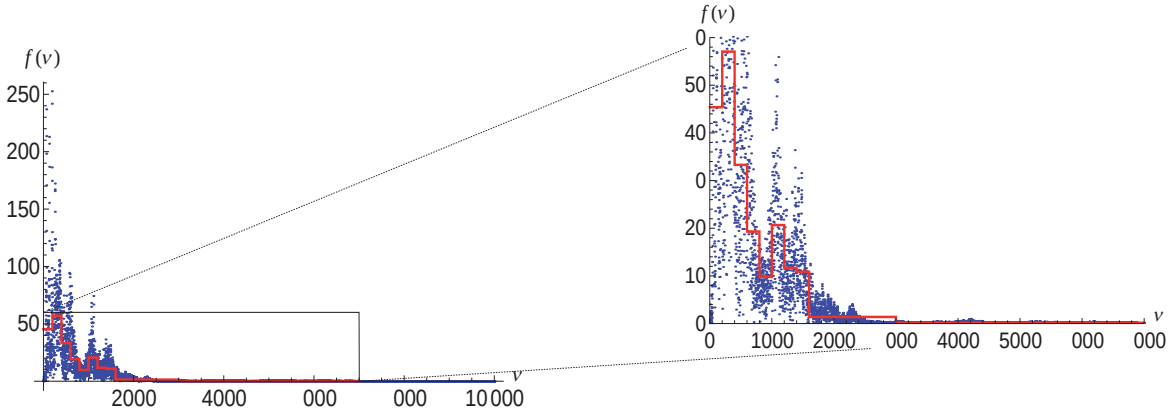

Figura 4.5: Promedios de las amplitudes pertenecientes a la palabra hola de la persona 1. Las amplitudes que se tomaron son $a_j = \sum_{i=-199+200j}^{200j} \frac{|f(\nu_i)|}{200}$ con $1 \leq j \leq 8$, $a_9 = \sum_{i=1601}^{3000} \frac{|f(\nu_i)|}{1400}$ y $a_{10} = \sum_{i=3001}^{7000} \frac{|f(\nu_i)|}{4000}$.

Lo que se hizo para tomar los datos para las diez neuronas en la capa de entrada, fue sacar el promedio de las amplitudes de grupos de 200 frecuencias, entre 1 y 1600 (es donde hay mas cambios en las amplitudes). Después se tomó el promedio de un grupo de 1400 frecuencias (de 1600 a 3000) y por último un promedio de un grupo de 4000 (de 3001 a 7000), esto se puede observar en la Figura 4.5. La información de frecuencias mas altas de 7000, no se toma en cuenta ya que prácticamente son cero. Se trató de tomar más muestras en donde hay frecuencias con mayores amplitudes, pero se pudo

haber utilizado cualquier otro criterio como veremos en la próxima sección. Una vez teniendo estas amplitudes se normalizaron en el rango de sensibilidad de la función de activación (se usó nuevamente la tangente hiperbólica) y la condición que se usó fue la siguiente:

$$\hat{a}_i = \frac{a_i}{50} \tag{4.2}$$

con i entero en $1 \leq i \leq 10$ obteniendo los valores mostrados en la Tabla 4.4. Con esta normalización se obtienen valores en el rango de sensibilidad de la función.

Persona 1		Persona 2		Persona 3	
a	$\hat{a}$	a	$\hat{a}$	a	$\hat{a}$
45.387	0.9077	12.015	0.2403	7.381	0.1476
57.06	1.1412	22.634	0.4526	16.992	0.3398
33.297	0.6659	7.53	0.1506	19.05	0.381
19.31	0.3862	12.214	0.2442	8.313	0.1662
9.782	0.1956	3.554	0.0710	11.039	0.2207
20.665	0.4133	5.556	0.1111	4.032	0.0806
11.668	0.2333	1.593	0.0318	4.086	0.0817
10.89	0.2178	2.002	0.0400	3.515	0.0703
1.387	0.0277	1.031	0.0206	1.429	0.0285
0.154	0.0030	0.085	0.0017	0.27	0.0054

Tabla 4.4: Valores del promedio de las amplitudes (a) con sus respectivas normalizaciones (â) de la palabra hola perteneciente a las personas 1 (masculino), 2 (masculino) y 3 (femenino).

Para ir calibrando los parámetros con los que el entrenamiento de la red es satisfactorio, primero fijamos la mayoía de ellos dejando libre únicamente la constante de aprendizaje. Empezando con el valor de $\gamma = 1$, se examina el comportamiento del error y si no se ve convergencia hacia ningÃºn valor se disminuye el valor de γ, a la mitad por ejemplo. Se vuelve a entrenar la red con este nuevo valor de γ y se verifica el error nuevamente. Si el error ahora resulta que casi no varía, es decir, se ve constante pero en un valor lejano de cero, es indicador de que γ es muy pequeño y hay que realizar

nuevamente el entrenamiento con un valor mayor de este parámetro, en caso contrario, se disminuye nuevamente. Así se va acotando el valor del parámetro de aprendizaje, hasta encontrar el que funciona mejor, para esto es necesario de 10 a 15 corridas. Si con el γ optimo se tiene aun un error grande, se vuelve a realizar el mismo procedimiento, pero ahora con un nuevo intervalo donde se inicializan aleatoriamente los pesos. Si después de ir probando con otros valores iniciales de los pesos vemos que el error no converge a ningún valor, es decir, oscila demasiado, es necesario aumentar el número de neuronas en la capa oculta, debido a que tal vez la información a clasificar es mucha y requiere de un mayor número de neuronas en la capa oculta para poder clasificarla correctamente. Si al encontrar un número adecuado de neuronas en la capa oculta (el error ya converge, aunque no sea a un valor muy cercano a cero) y realizar los pasos anteriores seguimos sin poder entrenar la red de manera adecuada, se puede proseguir a probar cambiando tanto los objetivos propuestos o la normalización de los datos de entrada. Por último, si nada de esto funciona, se hacen todas las pruebas anteriores pero cambiando la función de activación. Como podemos ver, al ser demasiados los parámetros involucrados, es conveniente ir complicando el problema poco a poco para así tener una idea de los parámetros que hay que modificar. Por ejemplo, al iniciar con solo 3 personas y con solo una palabra para clasificar, nos damos una idea del número de neuronas en la capa oculta necesarias y así al tratar de clasificar dos palabras, desde antes podemos saber que tal vez sea necesario aumentar el número de neuronas en esta capa. También al complicar el problema con mayor información a clasificar, se puede comenzar a explorar a partir del valor de γ que se usó al entrenar la red satisfactoriamente con menor cantidad de datos.

En los problemas donde se requirió mayor tiempo de cálculo para el entrenamiento de la red, se examinó el comportamiento del error con un número menor de iteraciones, ya que como veremos en la siguiente sección, para un entrenamiento con 100000 iteraciones, se requería de un tiempo de cálculo de aproximadamente dos horas para examinar una combinación de parámetros, por lo que se examinaba el comportamiento para 20000 iteraciones y si observábamos que el error no disminuía, se utilizaban otros parámetros, ya que en general si no se ve una tendencia de convergencia a cero en el error, ya no lo hará aunque se utilice un número mayor de iteraciones.

Una vez que tenemos la información que se va a introducir en la red, se modificaron los parámetros hasta encontrar los que nos sirvieron para que el error convergiera a cero y poder hacer la clasificación de las palabras, mostrándose los resultados en la Figura 4.6.

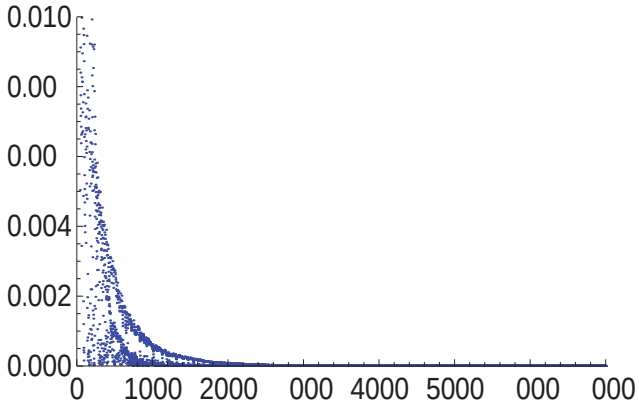

Figura 4.6: Disminución del error $E = \frac{1}{2}\sum_{k=1}^{salida}(t_k - s_k)^2$ al entrenar la red 7000 iteraciones con la palabra "hola" perteneciente a las 3 personas. Se utilizaron los siguientes parámetros: tangente hiperbólica como función de activación, 10 neuronas en la capa oculta, parámetro de aprendizaje $\gamma = .05$, bias $\theta = 0$, objetivos 0.09, 0.01 y los valores iniciales de los pesos entre [-1,1]. A partir de la iteración 1500 ya no disminuye considerablemente el error pero se quiso obtener valores más cercanos a los objetivos.

Los valores de las neuronas de salida una vez que el entrenamiento terminó y que tenemos los valores de los pesos actualizados para clasificar la información como deseamos, se pueden observar en la Tabla 4.5.

Después de haber probado la red y saber que se obtienen buenos resultados con los parámetros establecidos, se modificó la misma para incluir el término de momento, que es el último parámetro que nos falta por examinar. Al variar los parámetros encontramos que la constante del momento sirve para hacer que en ocasiones el error converja a cero, cuando éste oscila (Figura 4.7). Se produce el mismo efecto que se genera al reducir el parámetro de aprendizaje como vimos anteriormente. Al menos que al re-

Neurona	Persona 1	Persona 2	Persona 3
s_1	0.09	0.01	0.01
s_2	0.01	0.09	0.0099
s_3	0.01	0.01	0.09

Tabla 4.5: Valores de las neuronas de salida al introducir las amplitudes normalizadas de la palabra "hola" para cada persona, una vez que se entrenó la red. El número de neurona que tiene valor aproximado de 0.09 corresponde al número de persona a la que pertenece la voz.

ducir el parámetro de aprendizaje no lográramos que el error deje de oscilar, es cuando podemos utilizar el término de momento, por que si se usa cuando el error converge puede hacer que lo desestabilice.

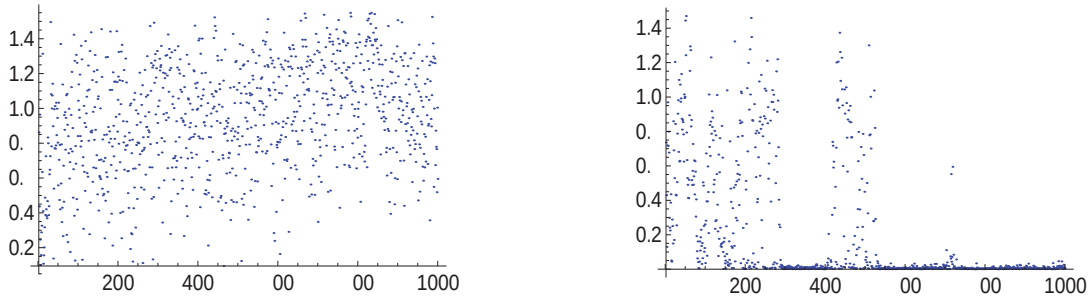

Figura 4.7: Disminución del error cuando oscila. Se utilizaron los siguientes parámetros: tangente hiperbólica como función de activación, 10 neuronas en la capa oculta, parámetro de aprendizaje $\gamma = .5$, bias $\theta = 0$, objetivos 0.09, 0.01 y los valores iniciales de los pesos entre [-1,1]. A la izquierda se usó $\mu = 0$. A la derecha $\mu = 0.6$. Se observa como mejora el comportamiento del error al incluir la constante del momento μ.

Es por esto que es no es conveniente usar la constante de momento, a menos que sea necesario y que se tenga idea de donde se pueden poner, ya que podemos obtener

el mismo efecto al disminuir el parámetro de aprendizaje.

Se comenzó a complicar poco a poco la red para introducir la voz de más personas y que además nos clasifique el género de la persona que esta hablando. Se utilizó la voz de tres personas más, en esta ocasión dos mujeres y un hombre, para tener un total de seis (tres hombres y tres mujeres) y se normalizaron los datos pertenecientes a la palabra "hola", nuevamente usando la normalización de la Ecuación 4.2 obteniendo los resultados de la Tabla 4.6.

Persona 4		Persona 5		Persona 6	
a	$\hat{a}$	a	$\hat{a}$	a	$\hat{a}$
29.716	0.5943	29.047	0.5809	14.704	0.2940
6.899	0.1379	34.077	0.6815	30.664	0.6132
2.165	0.0433	18.835	0.3767	14.4945	0.2898
0.419	0.0083	5.5114	0.1102	10.3	0.206
0.678	0.0135	6.945	0.1389	3.018	0.0603
0.559	0.0111	2.302	0.0460	2.555	0.0511
1.01	0.0202	2.294	0.0458	6.965	0.1393
0.546	0.0109	2.916	0.0583	2.683	0.0536
0.135	0.0027	0.994	0.0198	1.536	0.0307
0.031	0.0006	0.316	0.0063	0.184	0.0036

Tabla 4.6: Valores del promedio de las amplitudes (a) con sus respectivas normalizaciones ($\hat{a}$) de la palabra hola perteneciente a las personas 4 (masculino), 5 (femenino) y 6 (femenino).

Adicionalmente se agregaron otras dos neuronas en la capa de salida que servirán para indicarnos el sexo de la persona. Para activar estas últimas neuronas se impuso la condición de que si

$$\hat{a}_{10} > .0035 \tag{4.3}$$

correspondería a la activación de la octava neurona en la capa de salida y que nos indica que el sexo es femenino, de lo contrario, se activa la séptima neurona indicando que el sexo es masculino. Esto se hizo así debido a que las mujeres tienen por lo general

voces mas agudas a los hombres, lo que nos indica que las frecuencias altas de las palabras tendrán una mayor amplitud comparada con la de los hombres y es por eso que en se impone esta condición en la ultima amplitud normalizada de la palabra, que correspondería a las frecuencias más altas. Se puede observar este hecho al comparar la última amplitud normalizada de los hombres correspondientes a las personas 1,2 y 4 con la de las mujeres que corresponden a las personas 3, 5 y 6. Se emplearon entonces 15 neuronas en la capa oculta (ya que con 10 como se había estado haciendo no era suficiente para la convergencia del error) y 8 en la capa de salida, variando los diferentes parámetros hasta encontrar los que dieron como resultado los objetivos deseados al introducir las amplitudes normalizadas una vez que se entrenó (Figura 4.8).

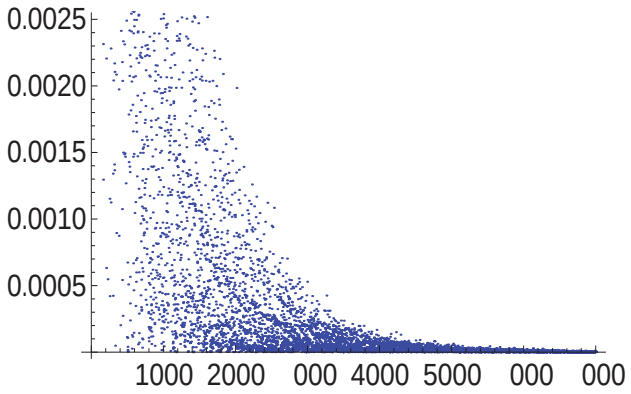

Figura 4.8: Disminución del error $E = \frac{1}{2}\sum_{k=1}^{salida}(t_k - s_k)^2$ para un total de 7000 iteraciones en el entrenamiento de la palabra hola incluyendo detección del género de la persona. Los parámetros usados son: tangente hiperbólica como función de activación, valores iniciales de los pesos entre [-1,1], objetivos iguales a $\{0.09, 0.01\}$, $\gamma = 0.1$, $\theta = 0$ y $\mu = 0$.

Los valores que se obtienen en las unidades de salida al usar las matrices de los pesos que resultaron del proceso de entrenamiento se pueden observar en la Tabla 4.7. Aunque los valores que se obtienen no son exactamente los propuestos, ya son útiles para identificar a la persona y el genero de la misma.

Neurona	Persona 1	Persona 2	Persona 3	Persona 4	Persona 5	Persona 6
s_1	0.0898	0.0082	0.0090	0.0094	0.0101	0.0105
s_2	0.0084	0.0858	0.0081	0.0093	0.0092	0.0117
s_3	0.0105	0.0102	0.0898	0.0097	0.0109	0.0098
s_4	0.0111	0.0118	0.0107	0.0900	0.0107	0.0090
s_5	0.0092	0.0078	0.0089	0.0095	0.0897	0.0108
s_6	0.0107	0.0119	0.0109	0.0103	0.0103	0.0892
s_7	0.0888	0.0874	0.0090	0.0897	0.0090	0.0110
s_8	0.0092	0.0091	0.0898	0.0102	0.0891	0.0906

Tabla 4.7: Valores de las neuronas de salida al introducir las amplitudes normalizadas de la palabra "hola" dicha por cada persona, después del entrenamiento de la red. El número de neurona entre 1 y 6 con valor aproximado a 0.09 corresponde al número de la persona. El valor aproximado a 0.09 en la neurona 7 indica que el sexo de la persona es masculino. El valor aproximado a 0.09 en la neurona 8 indica que el sexo de la persona es masculino.

No se utilizó un mayor número de iteraciones por que el tiempo de entrenamiento ya empezaba a incrementarse considerablemente con 15 neuronas en la capa oculta y además es solo una palabra, por lo que al tratar de entrenar las demás palabras a la vez se incrementará aún mas el tiempo que requiere el entrenamiento. Más adelante optimizaremos la forma de clasificar los objetivos para utilizar un menor número de neuronas en la capa de salida.

Ya hemos visto que al utilizar un promedio de las amplitudes, nos puede llevar a un entrenamiento satisfactorio de la red, pero antes de introducir las demás palabras que se grabaron, pasaremos a usar las frecuencias en vez de las amplitudes para el mismo fin. Esto es por que si usamos las amplitudes y quisiéramos hacer una prueba de reconocimiento usando otra grabación con la misma palabra y la persona que la dijo, debería de decirla con la misma intensidad o al menos muy cercana a la que dijo en la primera grabación, por lo que deberíamos de hacer otras modificaciones a la señal y queremos ver si es posible hacer que funcione la red utilizando las frecuencias, sin necesidad de hacer mas modificaciones.

4.3 RNA para reconocimiento de voz usando frecuencias

Abordaremos ahora el problema de reconocimiento de voz usando ciertas frecuencias del espectro en lugar de un promedio de las amplitudes. Las ideas serán las mismas que se han usado hasta ahora en cuanto al número de neuronas en la capa de entrada (se utilizarán 10), el número de neuronas en la capa oculta (nuevamente se iniciarán las pruebas con 10 neuronas por simplicidad y se aumentarán conforme sea necesario) y el número de neuronas en la capa de salida que dependerá del número de personas. Queremos además que la red nos indique el sexo de la persona a la que pertenece la voz, por lo que necesitaremos dos neuronas más en la capa de salida que nos ayudará con esta labor y de la que hablamos con mayor detalle más adelante. Los valores de los objetivos que nos indicarán la persona y el género serán nuevamente de 0.09 y .01, que hasta el momento no hemos tenido problema con estos valores, pero de ser necesario se tomarán cualquier otro valor (con las características mencionadas en las secciones anteriores). Para obtener los diez valores de entrada, lo que se realizó fue desglosar el espectro de frecuencias que se obtuvo al hacer la FFT, en bloques de cien hasta un valor en el que los datos obtenidos ya no "aporten información fundamental" de la palabra. Con esto nos referimos a que, por ejemplo, en el caso de la palabra "hola" solo se consideraron las frecuencias menores a 3000, debido a que el valor de las amplitudes para frecuencias mayores a 3000 eran prácticamente cero. De cada uno de estos bloques que contienen cien frecuencias, se consideró solo la frecuencia que tiene la mayor amplitud dentro de cada bloque. Por último, del total de 30 frecuencias que tomamos, se seleccionaron las diez con mayor amplitud una vez mas. (Figura 4.9). Una vez que se eligieron las diez frecuencias, se normalizaron con valores en los que la función de activación (la tangente hiperbólica, por que hasta el momento es con la que se ha estado trabajando sin problemas) es más sensible y la fórmula que se usó es:

$$\xi_i = \left(\nu_i - \sum_{J=1}^{10} \frac{\nu_j}{10}\right)\frac{1}{1000} + \frac{i}{4} - \frac{3}{2} \tag{4.4}$$

En donde ν_i es la frecuencia i-ésima para cada persona, ξ_i es la correspondiente frecuencia normalizada e i es un entero tal que $1 \leq i \leq 10$. Se utilizaron en un principio frecuencias normalizadas con valores entre cero y dos (dividiendo simplemente las fre-

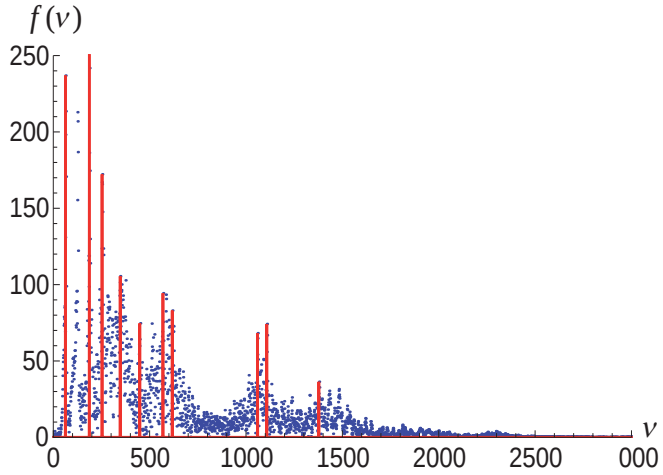

Figura 4.9: Espectro de frecuencias para la palabra "hola" de la persona 1 en donde las líneas indican las diez frecuencias que fueron seleccionadas para posteriormente aplicarles una normalización.

Persona 1		Persona 2		Persona 3	
ν	ξ	ν	ξ	ν	ξ
63	-1,08	59	-1,10	138	-0,98
187	-0,83	118	-0,99	245	-0,76
253	-0,70	294	-0,63	367	-0,52
347	-0,51	343	-0,54	421	-0,41
449	-0,30	414	-0,39	558	-0,14
569	-0,06	594	-0,03	603	-0,05
617	0,02	678	0,13	725	0,24
1060	0,91	744	0,26	880	0,50
1108	1,00	1118	1,01	968	0,67
1378	1,54	1768	2,31	1361	1,46

Tabla 4.8: Valores de las frecuencias (ν) con sus respectivas normalizaciones (ξ) de la palabra "hola" perteneciente a las personas 1 (masculino), 2 (masculino) y 3 (femenino).

cuencias entre mil) pero lo que dió mejor resultado fue utilizar un intervalo mas amplio entre [-2,2]. La normalización de los datos de la ecuación anterior para la palabra "hola" de las tres personas se observa en Tabla 4.8. Con esta normalización y variando los parámetros para que el error durante el entrenamiento de la red convergiera a cero, se obtuvieron los resultados para el error entre los valores propuestos y los valores obtenidos al propagar la información, mostrados en la Figura 4.10.

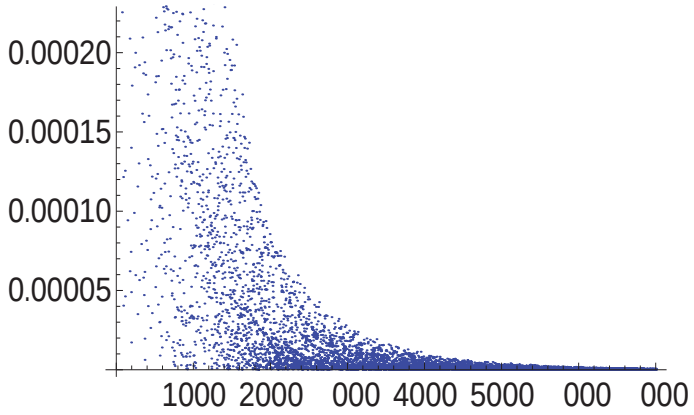

Figura 4.10: Disminución del error $E = \frac{1}{2}\sum_{k=1}^{salida}(t_k - s_k)^2$ para 7000 iteraciones en el entrenamiento de la red para la palabra "hola" usando las frecuencias normalizadas. Parámetros utilizados: función tangente hiperbólica, 10 neuronas en la capa oculta, valores objetivo de $\{.09, .01\}$, pesos aleatorios iniciales entre $[-1, 1]$, $\gamma = 0.05$, $\theta = 0$ y $\mu = 0$.

Ya que se entrenó la red y se actualizaron los pesos de tal manera que al introducir los datos iniciales, en este caso los números que caracterizan la palabra "hola", nos resulten los valores deseados. Para este ejemplo, los valores que resultaron en la capa con las unidades de salida, son los que se muestran en la Tabla 4.9. Comenzamos únicamente con tres personas para encontrar las condiciones iniciales adecuadas, como lo son, el valor de los pesos con los que se inicializa la red, la normalización y los objetivos. Ahora se quiere incrementar la cantidad de personas que se usa para el reconocimiento de la voz a un total de seis (tres del sexo masculino y tres del sexo femenino) y añadir características a la red como la capacidad de identificar el sexo de

Neurona	Persona 1	Persona 2	Persona 3
s_1	0.0893	0.0084	0.0135
s_2	0.0098	0.0908	0.0113
s_3	0.0102	0.0098	0.089

Tabla 4.9: Valores de las neuronas de salida al introducir las frecuencias normalizadas pertenecientes a la palabra "hola" de las personas 1, 2 y 3 después de entrenar la red. El número de neurona que tiene valor aproximado de 0.09 corresponde al número de persona a la que pertenece la voz.

la persona y tomar en cuenta todas las palabras que se grabaron previamente.
Se utilizó la misma estructura que en el caso anterior para las amplitudes, en donde se usaron 8 neuronas en la capa de salida (6 para indicar que persona dijo la palabra y 2 para indicar el sexo de la persona). En este punto se pensó que solo faltaría modificar el numero de neuronas en la capa oculta y ajustar el parámetro de aprendizaje como sucedió cuando se tomaron las amplitudes, pero desafortunadamente no fue así. A pesar de que se hicieron muchos intentos modificando todos los parámetros no se logró entrenar correctamente a la red con la normalización utilizada antes (Ecuación 4.4). Así que después de intentar con distintas normalizaciones, parámetros y cambiar los valores de los objetivos a 0.9 si se activa o 0.1 si no es así, se encontró una normalización con la que se entrenó satisfactoriamente a la red:

$$\xi_i = \left(\nu_i - \sum_{J=1}^{10} \frac{\nu_j}{10}\right)\frac{1}{500} \tag{4.5}$$

con $1 \leq i \leq 10$. Los nuevos valores de las frecuencias normalizadas para cada persona se encuentran en la Tabla 4.10. Al usar las nuevas frecuencias normalizadas como datos de entrada para el entrenamiento de la red y buscar los parámetros óptimos a base de ensayo y error, se encontró el comportamiento del error para cada una de las iteraciones que se pueden ver en la Figura 4.11. Los valores de las neuronas de salida al introducir las frecuencias normalizadas correspondientes a la palabra "hola" de las 6 personas, después de que se adaptaron los pesos durante el entrenamiento, se observan en la Tabla 4.11.

Para identificar el sexo de la persona se recurrió al método usado en el caso en que

Persona 1		Persona 2		Persona 3		Persona 4		Persona 5		Persona 6	
ν	ξ	ν	ξ	ν	ξ	ν	ξ	ν	ξ	ν	ξ
63	-1,08	59	-1,10	138	-0,98	40	-1,11	128	-0,99	100	-0,94
187	-0,83	118	-0,99	245	-0,76	164	-0,86	235	-0,78	107	-0,93
253	-0,70	294	-0,63	367	-0,52	224	-0,74	350	-0,55	214	-0,71
347	-0,51	343	-0,54	421	-0,41	332	-0,53	402	-0,44	320	-0,50
449	-0,30	414	-0,39	558	-0,14	468	-0,25	583	-0,08	426	-0,29
569	-0,06	594	-0,03	603	-0,05	533	-0,12	701	0,14	533	-0,08
617	0,02	678	0,13	752	0,24	602	0,00	721	0,18	638	0,12
1060	0,91	744	0,26	880	0,50	980	0,76	861	0,46	744	0,34
1108	1,00	1118	1,01	968	0,67	1278	1,36	929	0,60	1278	1,40
1378	1,54	1768	2,31	1361	1,46	1350	1,50	1354	1,45	1379	1,61

Tabla 4.10: Valores de las frecuencias (ν) seleccionadas para las seis personas pertenecientes a la palabra "hola", usando las normalizaciones (ξ) obtenidas con la Ecuación 4.5.

Neurona	Persona 1	Persona 2	Persona 3	Persona 4	Persona 5	Persona 6
s_1	0.8795	0.1303	0.1316	0.1174	0.1303	0.1160
s_2	0.1083	0.8997	0.1064	0.1051	0.1068	0.1044
s_3	0.1042	0.1046	0.8772	0.1022	0.1032	0.1014
s_4	0.1087	0.9124	0.0917	0.8760	0.9106	0.1083
s_5	0.0910	0.0914	0.0878	0.0932	0.8841	0.0949
s_6	0.0805	0.0859	0.0887	0.0819	0.0894	0.8904

Tabla 4.11: Valores de las neuronas de salida al introducir las frecuencias normalizadas correspondientes a la palabra "hola" de las 6 personas. El número de neurona que tiene valor aproximado de 0.9 corresponde al número de persona a la que pertenece la voz.

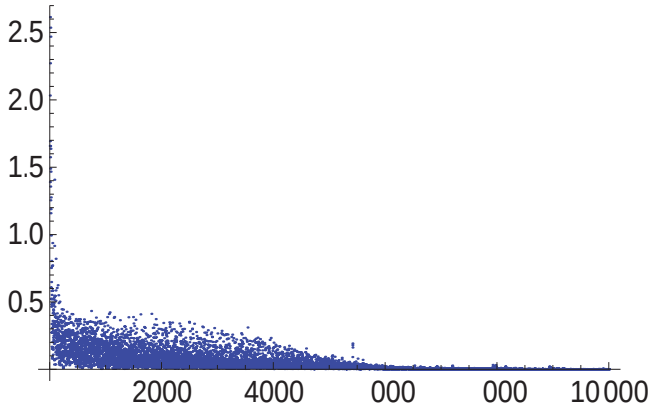

Figura 4.11: Disminución del error $E = \frac{1}{2}\sum_{k=1}^{salida}(t_k - s_k)^2$ para 10000 iteraciones en el entrenamiento de la red usando las frecuencias normalizadas correspondientes a la palabra "hola" de las 6 personas. Parámetros: Función tangente hiperbólica, 15 neuronas en la capa oculta, $\gamma = .05$, $\theta = 0$, $\mu = 0$, pesos aleatorios iniciales entre [-1,1] y valores objetivo de $\{.9, .1\}$.

se tomaron los promedios de las amplitudes, identificando frecuencias mas bajas para las palabras habladas por los hombres (Personas 1,2 y 4) y frecuencias mas altas en el caso de las mujeres (Personas 3,5 y 6). La condición impuesta es que si se cumple

$$\xi_1 < -1.00 \tag{4.6}$$

entonces se activa la séptima neurona con el valor 0.9 indicando que es hombre y la octava neurona con el valor de 0.1, en caso contrario a la condición impuesta, se activa la neurona octava con el valor 0.9 indicando que es mujer y la séptima neurona toma el valor de 0.1.

Utilizando los mismos parámetros para cuando no incluimos el sexo de la persona y usando 10000 iteraciones para el entrenamiento de la red se llegó al resultado de la Figura 4.12, en donde se observa como el error va disminuyendo con cada paso. En la Tabla 4.12 se observa, una vez que se entrenó la red, los valores de las neuronas de salida al introducir las frecuencias normalizadas correspondientes a la palabra "hola" de las 6 personas y que incluyen las dos neuronas asociadas a la identificación del sexo.

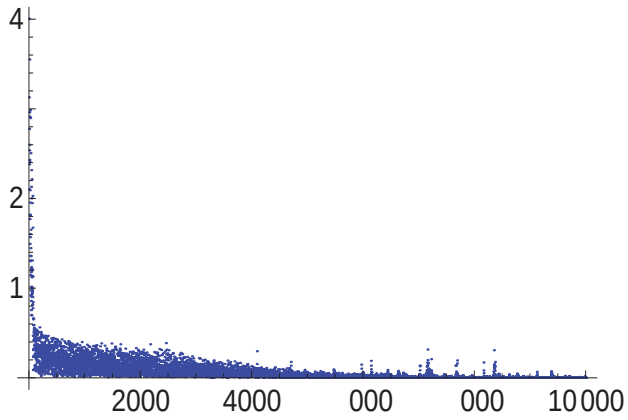

Figura 4.12: Disminución del error $E = \frac{1}{2}\sum_{k=1}^{salida}(t_k - s_k)^2$ para 10000 iteraciones en el entrenamiento de la red para la palabra "hola" incluyendo la detección del sexo de la persona. Parámetros: Función tangente hiperbólica, 15 neuronas en la capa oculta, $\gamma = .05$, $\theta = 0$, $\mu = 0$, pesos aleatorios iniciales entre [-1,1] y valores objetivo de $\{.9, .1\}$.

Neurona	Persona 1	Persona 2	Persona 3	Persona 4	Persona 5	Persona 6
s_1	0.8622	0.1128	0.0834	0.0604	0.0855	0.0766
s_2	0.0942	0.9036	0.0920	0.0971	0.0905	0.0924
s_3	0.1421	0.1063	0.8861	0.1238	0.1383	0.1018
s_4	0.1330	0.0837	0.1068	0.8838	0.1009	0.1202
s_5	0.0721	0.0908	0.0841	0.0894	0.8804	0.1043
s_6	0.0868	0.1249	0.1048	0.0842	0.1098	0.8890
s_7	0.9732	0.8929	0.1055	0.8828	0.1052	0.1171
s_8	0.1184	0.1143	0.9111	0.1274	0.9361	0.8891

Tabla 4.12: Valores de las neuronas de salida al introducir las frecuencias normalizadas correspondientes a la palabra "hola" de las 6 personas. El número de neurona entre 1 y 6 con valor aproximado a 0.9 corresponde al número de la persona. El valor aproximado a 0.9 en la neurona 7 indica que el sexo de la persona es masculino. El valor aproximado a 0.9 en la neurona 8 indica que el sexo de la persona es masculino.

Se puede ver en los resultados, como los valores que se calculan se aproximan en gran medida a los valores propuestos. Los datos que corresponden a la palabra de la persona 1, la cual es del sexo masculino se aproximan a los valores de salida propuestos para este caso:

$$\{\ 0.9\ ,\ 0.1\ ,\ 0.1\ ,\ 0.1\ ,\ 0.1\ ,\ 0.1\ ,\ 0.9\ ,\ 0.1\ \}$$

$$\Downarrow$$

$$\{\ 0.8622\ ,\ 0.0942\ ,\ 0.1421\ ,\ 0.1330\ ,\ 0.0721\ ,\ 0.0868\ ,\ 0.9732\ ,\ 0.1184\ \}$$

en donde la neurona que se "activa" (el elemento de la lista) con el valor de 0.9, corresponde al número de la persona a la que pertenece la voz (entre 1 y 6), el cual corresponde al primer elemento en este caso y las demás neuronas permanecen "desactivadas" con el valor de 0.1. También se activa la séptima neurona, debido a que en esta situación la persona es un hombre.

Una vez que se entrenó la red para detectar quién de las 6 personas dijo la palabra "hola" al introducir las frecuencias normalizadas y también detectar el sexo de la persona, se procedió a introducir las demás palabras para ver como funciona la red con las normalizaciones y parámetros antes establecidos (función de activación tangente hiperbólica, $\theta = 0$, $\mu = 0$, 15 neuronas en la capa oculta, valores iniciales de los pesos [-1,1] y valores de los objetivos 0.9 y 0.1). Se aplicó el mismo proceso de selección de frecuencias a las palabras "bienvenido" (Figura 4.13), "adiós" (Figura 4.14) y "felicidades" (Figura 4.15) que se le aplicó a la palabra "hola". Por el momento no se considerará la detección de sexo de la persona para estas palabras, debido a que la forma en que se tomaron los datos de entrada, no reflejan frecuencias mas altas para las mujeres y menores para los hombres como sucedió con la palabra "hola" (habría que seleccionar de manera diferente las frecuencias que se introducen en la red para cada palabra, tarea que se realizará si el tiempo es suficiente). Por este motivo se tomarán únicamente 6 neuronas en la capa de salida, descartando las que se utilizaron anteriormente para indicar el sexo. Tampoco se considerará la voz grabada de la palabra con el nombre de cada persona, ya que éste es diferente para cada persona. Sólo se muestra el espectro de las palabras de la persona 1 para representar como se tomaron las frecuencias, pero el espectro de las palabras para las demás personas y la selección es similar.

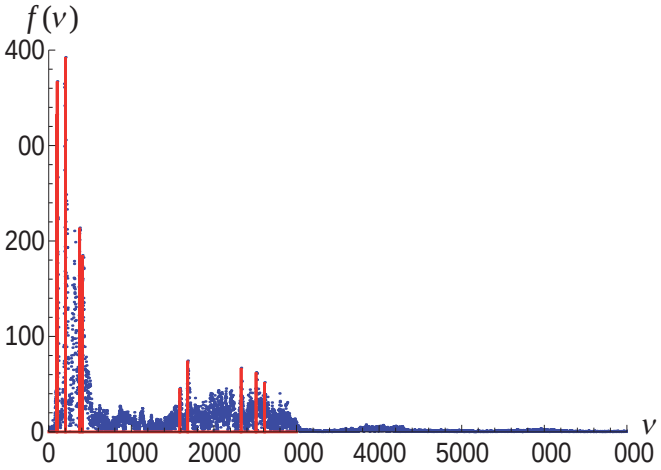

Figura 4.13: Espectro de frecuencias para la palabra "bienvenido" de la persona 1. Las líneas indican el conjunto de 10 frecuencias que se seleccionaron para después normalizarlas.

Persona 1		Persona 2		Persona 3		Persona 4		Persona 5		Persona 6	
ν	ξ	ν	ξ	ν	ξ	ν	ξ	ν	ξ	ν	ξ
100	-2.18	87	-2.16	44	-1.51	90	-1.93	508	-1.44	164	-1.46
102	-2.17	174	-1.98	188	-1.22	184	-1.74	200	-1.14	288	-1.21
203	-1.97	264	-1.80	201	-1.20	277	-1.55	396	-0.75	353	-1.08
375	-1.63	354	-1.62	401	-0.80	350	-1.41	4038	-0.74	432	-0.92
408	-1.56	417	-1.50	407	-0.79	436	-1.23	595	-0.35	529	-0.73
1588	0.79	1500	0.665	541	-0.52	525	-1.06	603	-0.34	701	-0.38
1683	0.98	1501	0.667	627	-0.35	1583	1.05	712	-0.12	706	-0.37
2329	2.27	1735	1.13	1443	1.28	2020	1.92	802	0.05	878	-0.03
2510	2.63	2745	3.15	2001	2.39	2113	2.11	1402	1.25	1740	1.69
2614	2.84	2897	3.45	2171	2.73	2977	3.84	2586	3.62	3157	4.52

Tabla 4.13: Valores de las frecuencias (ν) seleccionadas para la palabra "bienvenido" con sus normalizaciones (ξ) para las 6 personas.

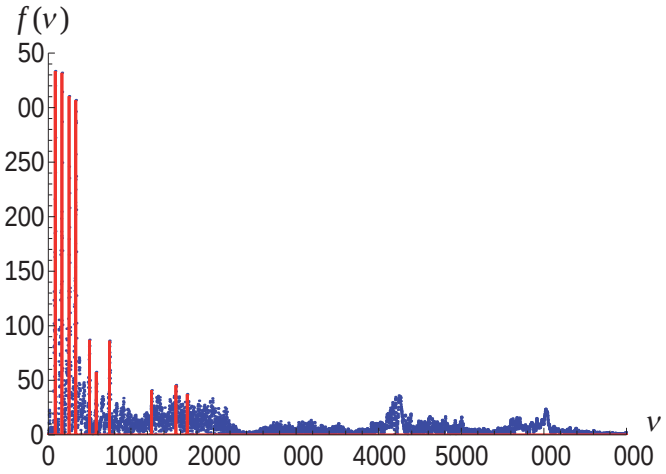

Figura 4.14: Espectro de frecuencias para la palabra "adiós" de la persona 1. Las líneas indican el conjunto de 10 frecuencias que se seleccionaron para después normalizarlas.

Persona 1		Persona 2		Persona 3		Persona 4		Persona 5		Persona 6	
ν	ξ	ν	ξ	ν	ξ	ν	ξ	ν	ξ	ν	ξ
83	-1.25	67	-1.13	146	-1.01	88	-1.72	175	-1.14	30	-0.66
165	-1.09	135	-0.99	293	-0.72	176	-1.55	202	-1.08	151	-0.41
251	-0.92	204	-0.85	301	-0.70	264	-1.37	353	-0.78	301	-0.28
333	-0.75	331	-0.60	411	-0.48	354	-1.19	4668	-0.56	304	-0.12
498	-0.42	403	-0.46	546	-0.21	410	-1.08	530	-0.43	486	0.06
579	-0.26	852	0.43	685	0.06	526	-0.85	620	-0.25	504	0.40
740	0.05	921	0.57	733	0.15	1751	1.59	770	0.04	618	0.41
1251	1.07	1006	0.74	821	0.33	1838	1.77	1081	0.66	705	0.76
1545	1.66	1201	1.13	1235	1.16	2012	2.11	1237	0.98	817	2.48
1681	1.93	1212	1.15	1374	1.43	2110	2.31	2035	2.57	1045	5.32

Tabla 4.14: Valores de las frecuencias (ν) seleccionadas para la palabra "adiós" con sus normalizaciones (ξ) para las 6 personas.

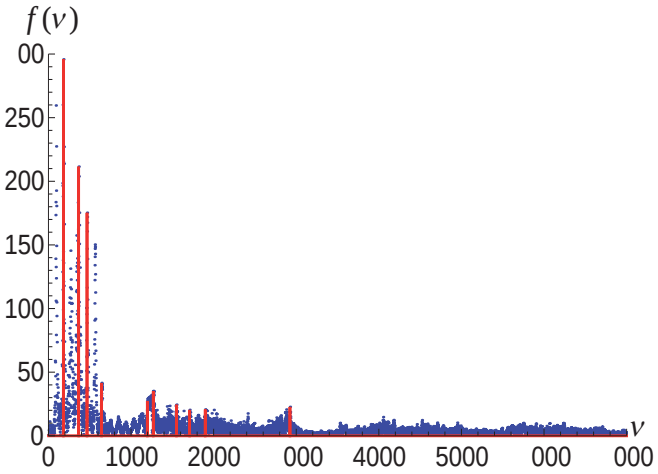

Figura 4.15: Espectro de frecuencias para la palabra "felicidades" de la persona 1. Las líneas indican el conjunto de 10 frecuencias que se seleccionaron para después normalizarlas.

Persona 1		Persona 2		Persona 3		Persona 4		Persona 5		Persona 6	
ν	ξ	ν	ξ	ν	ξ	ν	ξ	ν	ξ	ν	ξ
89	-1.14	82	-1.49	220	-1.42	96	-1.04	17	-1.19	70	-1.18
181	-0.95	189	-1.27	416	-1.02	190	-0.85	103	-1.02	185	-0.95
266	-0.78	229	-1.19	520	-0.82	286	-0.66	254	-0.72	201	-0.92
366	-0.58	334	-0.98	626	-0.60	381	-0.47	3078	-0.61	384	-0.55
467	-0.38	434	-0.78	742	-0.37	478	-0.27	495	-0.23	419	-0.48
562	-0.19	506	-0.64	843	-0.17	572	-0.09	550	-0.12	566	-0.19
643	-0.03	1294	0.93	923	-0.01	666	0.09	607	-0.01	625	-0.07
1201	1.08	1360	1.06	1059	0.25	763	0.29	802	0.37	754	0.18
1269	1.21	1901	2.14	1903	1.94	1326	1.41	1040	0.85	1521	1.72
1550	1.78	1942	2.22	2050	2.23	1421	1.60	1974	2.71	1885	2.44

Tabla 4.15: Valores de las frecuencias (γ) seleccionadas para la palabra "felicidades" con sus normalizaciones (ξ) para las 6 personas.

Tomando estas consideraciones se usó nuevamente la Ecuación 4.4 para normalizar

las frecuencias seleccionadas de las palabras (Tablas 4.13, 4.14, 4.15). Después de que tenemos la información que utilizaremos en el entrenamiento de la red, se pasó al entrenamiento de la misma, usando palabra por palabra. Se pensó que se tendría que usar una normalización diferente en cada palabra, pero lo sorprendente es que con esta normalización fue posible encontrar parámetros que se adaptaron para entrenar correctamente la red (para la palabra "felicidades" necesitó un entrenamiento con 15000 iteraciones en lugar de 10000). El comportamiento del error para el entrenamiento de cada una de las palabras se muestra en las Figuras 4.16, 4.17, 4.18, mientras que los resultados numéricos obtenidos en las neuronas de salida se muestran en las Tablas 4.16, 4.17, 4.18.

Hasta este momento lo que hemos logrado es, poder identificar cual de las 6 personas dijo cada palabra utilizando el entrenamiento de la red por separado, es decir, se entrenó cada palabra individualmente. Lo que se quiere realizar ahora es utilizar un solo entrenamiento de la red, para reconocer todas las palabras a la vez y nos indique cual de las personas la dijo. El proceso a seguir es el mismo que hemos hecho hasta ahora, solo que incluiremos la información de las cuatro palabras para cada persona en el entrenamiento de la red. La información que se introducirá en la red viene dada en una lista que contiene 6 vectores (o listas), cada uno de los cuales representa a cada una de las personas. A su vez cada uno de estos 6 vectores contiene 4 vectores, los cuales incluyen la información de las 10 frecuencias tomadas para cada palabra y por lo que estos vectores entonces representan las 4 palabras. Recordemos que para el entrenamiento de la red, la información debe introducirse de manera aleatoria para obtener los objetivos planteados. Entonces primero se escoge aleatoriamente un número entre 1 y 6, que nos indica a que persona pertenece la palabra, para posteriormente a seleccionar un número aleatorio entre 1 y 4, que representará a alguna de las palabra que contienen los 10 valores que se introducirán en las neuronas de la capa de entrada de la red. Se fue introduciendo palabra por palabra y se observó el comportamiento, primero con dos palabras, luego con tres y por último con cuatro. El problema que surgió al incrementar las palabrás o mejor dicho, la cantidad de datos,es que se requiere de un número mayor de iteraciones en el entrenamiento de la red para adaptar los pesos con la información adicional.

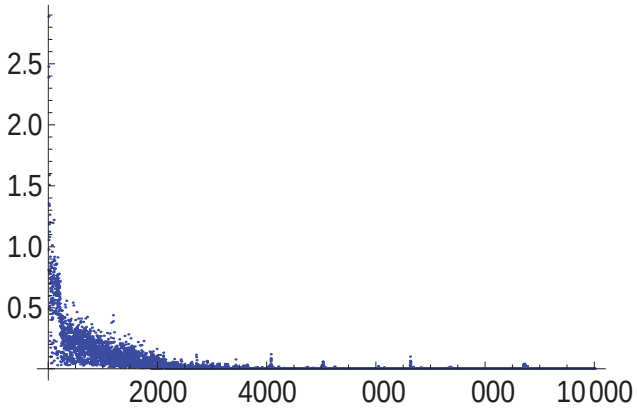

Figura 4.16: Disminución del error $E = \frac{1}{2}\sum_{k=1}^{salida}(t_k - s_k)^2$ para 10000 iteraciones en el entrenamiento de la red para la palabra "bienvenido". Parámetros: Función tangente hiperbólica, 15 neuronas en la capa oculta, $\nu = .05$, $\theta = 0$, $\mu = 0$, pesos aleatorios iniciales entre [-1,1] y valores objetivo de $\{.9, .1\}$.

Neurona	Persona 1	Persona 2	Persona 3	Persona 4	Persona 5	Persona 6
s_1	0.8986	0.0980	0.0997	0.1004	0.0996	0.1002
s_2	0.1004	0.9004	0.1010	0.1008	0.1017	0.1012
s_3	0.0994	0.1011	0.8995	0.0974	0.0975	0.0991
s_4	0.1004	0.1029	0.0982	0.8960	0.0997	0.1016
s_5	0.1006	0.1002	0.0962	0.0977	0.8951	0.0934
s_6	0.0988	0.0979	0.1008	0.0994	0.1061	0.8955

Tabla 4.16: Valores de las neuronas de salida al introducir las frecuencias normalizadas correspondientes a la palabra "bienvenido" de las 6 personas. El número de neurona con valor aproximado a 0.9 indica el número de la persona a la que pertenece la voz.

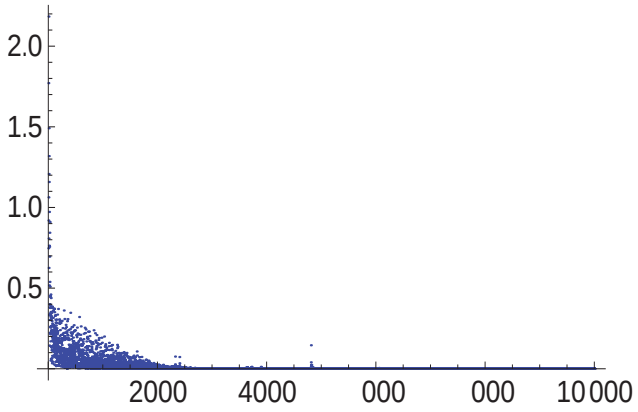

Figura 4.17: Disminución del error $E = \frac{1}{2}\sum_{k=1}^{salida}(t_k - s_k)^2$ para 10000 iteraciones en el entrenamiento de la red para la palabra "adiós". Parámetros: Función tangente hiperbólica, 15 neuronas en la capa oculta, $\gamma = .05$, $\theta = 0$, $\mu = 0$, pesos aleatorios iniciales entre [-1,1] y valores objetivo de $\{.9, .1\}$.

Neurona	Persona 1	Persona 2	Persona 3	Persona 4	Persona 5	Persona 6
s_1	0.9000	0.1000	0.1000	0.1000	0.1001	0.1002
s_2	0.0998	0.8998	0.8998	0.0999	0.1000	0.0999
s_3	0.1006	0.1005	0.1005	0.0999	0.1002	0.1002
s_4	0.1000	0.0998	0.0998	0.8990	0.0998	0.0997
s_5	0.0999	0.0999	0.0999	0.1000	0.9003	0.1000
s_6	0.0999	0.0998	0.0998	0.0998	0.0998	0.8999

Tabla 4.17: Valores de las neuronas de salida al introducir las frecuencias normalizadas correspondientes a la palabra "adiós" de las 6 personas. El número de neurona con valor aproximado a 0.9 indica el número de la persona a la que pertenece la voz.

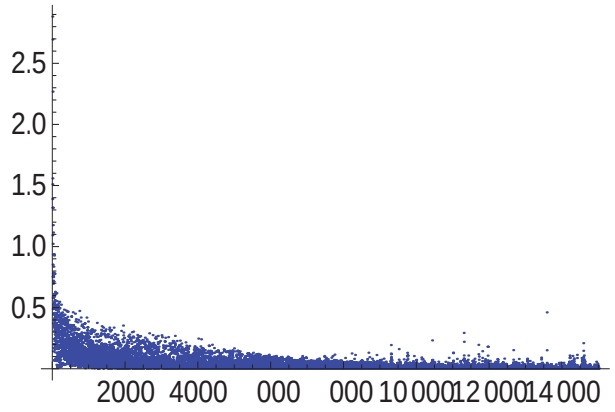

Figura 4.18: Disminución del error $E = \frac{1}{2}\sum_{k=1}^{salida}(t_k - s_k)^2$ para 15000 iteraciones en el entrenamiento de la red para la palabra "felicidades". Parámetros: Función tangente hiperbólica, 15 neuronas en la capa oculta, $\gamma = .05$, $\theta = 0$, $\mu = 0$, pesos aleatorios iniciales entre [-1,1] y valores objetivo de $\{.9, .1\}$.

Neurona	Persona 1	Persona 2	Persona 3	Persona 4	Persona 5	Persona 6
s_1	0.8778	0.1602	0.1229	0.1578	0.1771	0.1362
s_2	0.1014	0.8810	0.1054	0.0960	0.0665	0.0999
s_3	0.1631	0.1757	0.8272	0.1781	0.2000	0.2009
s_4	0.0532	0.0327	0.0706	0.8098	0.0082	0.0210
s_5	0.0812	0.0616	0.0898	0.0756	0.8835	0.0940
s_6	0.0854	0.0812	0.0807	0.0751	0.0863	0.8827

Tabla 4.18: Valores de las neuronas de salida al introducir las frecuencias normalizadas correspondientes a la palabra "felicidades" de las 6 personas. El número de neurona con valor más próximo a 0.9 indica el número de la persona a la que pertenece la voz.

Por este motivo el tiempo de cálculo se incrementa, ya que además de necesitar un número mayor de iteraciones, también se requiere en ocasiones incrementar el número de neuronas en la capa intermedia, dando como resultado un mayor número de cálculos. Se utilizaron los valores de los parámetros que se introdujeron cuando se entrenó la

palabra "hola", para entrenar al mismo tiempo las palabras "bienvenido" y "adiós" si. En principio se podría pensar que para el entrenamiento del conjunto de datos a procesar, de dos palabras, serían necesarias al menos el doble de iteraciones que cuando se entrenó solo una, debido a que es el doble de la información requerida para una palabra. Pero esto no sucedió así, ya que para el entrenamiento simultaneo de las palabras "hola" y "bienvenido", se requirió un total de 25000 iteraciones para adaptar los pesos, a diferencia de cuando se entrenó únicamente la palabra "hola" donde se necesitaron 10000 iteraciones. Después en conjunto con estas dos palabras, se seleccionó la palabra "adiós" y se requirieron para este caso un total de 50000 iteraciones para el entrenamiento de la red. Hasta este punto no se necesitó modificar los valores de los parámetros para el entrenamiento, solo se tuvo que aumentar el número de iteraciones. Por último para entrenar simultaneamente las 4 palabras se utilizaron 120000 iteraciones y se aumentó el número de neuronas en la capa oculta a 16. El error que se obtuvo durante el entrenamiento para cada iteración se muestra en la Figura 4.19, donde se puede observar que el error es mayor que cuando se entrenó cada palabra por separado, ya que es mas información la que se toma en cuenta en este proceso.

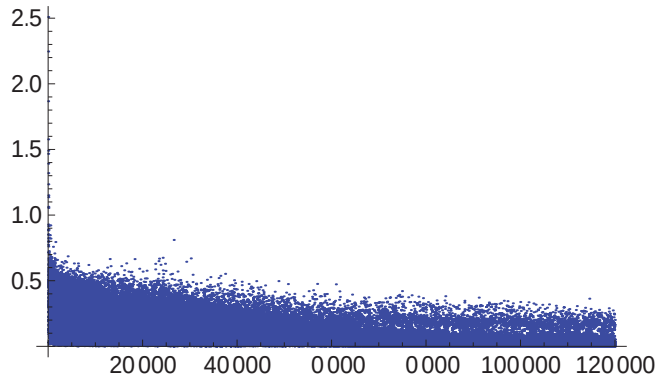

Figura 4.19: Disminución del error $E = \frac{1}{2}\sum_{k=1}^{salida}(t_k - s_k)^2$ para 120000 iteraciones en el entrenamiento simultaneo de la red para las palabras "hola", "bienvenido", "adiós" y "felicidades". Parámetros: Función tangente hiperbólica, 16 neuronas en la capa oculta, $\gamma = .05$, $\theta = 0$, $\mu = 0$, pesos aleatorios iniciales entre [-1,1] y valores objetivo de $\{.9, .1\}$.

Neurona	Persona 1	Persona 2	Persona 3	Persona 4	Persona 5	Persona 6
Palabra "hola"						
s_1	0.8632	0.1045	0.1108	0.1272	0.1235	0.1527
s_2	0.1291	0.8477	0.0932	0.1271	0.1061	0.1445
s_3	0.1379	0.1104	0.8427	0.0457	0.1297	-0.0217
s_4	0.1342	0.0890	0.1011	0.8724	0.1418	0.2936
s_5	0.1226	0.1086	0.0891	0.0791	0.8396	0.1114
s_6	0.1153	0.1005	0.0998	0.0905	0.1060	0.7587
Palabra "bienvenido"						
s_1	0.9185	0.1020	0.1546	0.1400	0.1347	0.1190
s_2	0.0878	0.8413	0.1340	0.0662	0.1083	0.0927
s_3	0.0797	0.1082	0.8941	0.1247	0.1119	0.0947
s_4	0.1324	0.1182	0.0795	0.9045	0.0924	0.1406
s_5	0.0726	0.0768	0.0929	0.0616	0.9196	0.0873
s_6	0.1010	0.1116	0.1303	0.1192	0.0951	0.9509
Palabra "adiós"						
s_1	0.8539	0.0840	0.1081	0.1046	0.1191	0.0927
s_2	0.1705	0.8712	0.1547	0.0880	0.1333	0.1522
s_3	0.1392	0.0903	0.7113	0.1050	0.1345	0.1316
s_4	0.1656	0.0988	0.1974	0.8986	0.1219	0.1212
s_5	0.0938	0.0846	0.0598	0.0968	0.8912	0.0543
s_6	0.1050	0.1192	0.0663	0.1011	0.1255	0.9425
Palabra "felicidades"						
s_1	0.9223	0.0960	0.0397	0.0701	0.0737	0.0739
s_2	0.0903	0.8299	0.1470	0.1980	0.0939	0.1253
s_3	0.0376	0.1123	0.7503	0.3406	0.1327	0.1220
s_4	0.0775	0.1462	0.2989	0.4575	0.0774	0.0362
s_5	0.0837	0.0829	0.1072	0.1048	0.9282	0.0720
s_6	0.1155	0.1040	0.0832	0.3092	0.0874	0.9307

Tabla 4.19: Valores de las neuronas de salida al introducir la información de las 4 palabras para las 6 personas después del entrenamiento de la red. La celda con mayor valor en cada columna representa el número de la persona que dijo cada palabra.

En la Tabla 4.19 se observan los valores de las neuronas de salida al propagar la información a través de la red con los pesos obtenidos después del entrenamiento. La celda con valor cercano a 0.9 (o en su defecto el valor mayor), corresponde al número de persona a la que pertenece la voz. Por ejemplo, después de introducir los 10 valores que representan la palabra "hola" de la persona 1, el valor mas grande es 0.8632 y se encuentra en la neurona de salida s_1, por lo que el número de la persona a la que pertenece la voz es la 1. Al ser un número mayor de datos que se entrenan, algunos resultados no son tan buenos como los propuestos como en el caso de la palabra "felicidades" para la persona 4, que aunque no se aproxima a 0.9 el valor de la neurona s_4, si es el mayor de todos y lo podemos identificar.

Al requerirse más tiempo durante el entrenamiento de la red con mayor número de palabras (o lo que es lo mismo, mayor cantidad de datos a clasificar), se incrementa demasiado el tiempo para encontrar los valores de los parámetros que hacen que funcione correctamente la red, ya que se necesitan ajustar varios parámetros en cada prueba para ver los que funcionan mejor. Para el entrenamiento de la red con las 4 palabras al mismo tiempo, el proceso requirió de 9136 segundos, que equivale aproximádamente a dos horas y media. Así que se optó por primero optimizar la red para que realice un menor número de cálculos, antes de incluir otras características. Lo que se hizo entonces fue disminuir la cantidad de neuronas en la capa de salida.

Con la idea usada hasta ahora de indicar la persona mediante un número representado por un 1 (0.9 o 0.09) en el elemento de la lista que corresponde al número de la persona y 0 (0.1 o 0.01) en las demás entradas, se requiere un total de neuronas en la capa de salida igual al número de personas a clasificar. Lo que usaremos ahora para disminuir el número de neuronas en la capa de salida, es la manera en que identificaremos o clasificaremos los datos que se introducen en la red. Para clasificar los datos se utilizará ahora una representación binaria dependiendo del número decimal asignado a la persona a la que pertenece la voz, es decir, si queremos clasificar a la persona número 3, se usará su representación binaria que en este caso serían dos valores: {1, 1}. Por ejemplo si tenemos dos personas, a una la llamamos persona 0 y a la otra persona 1, con una neurona en la capa de salida podemos clasificar los datos en 2 grupos. Si los

datos que representan la palabra de la persona cero son introducidos en la red, se utiliza como valor objetivo el 0. En caso contrario, si se introducen los datos de la persona 1 se propone un objetivo de 1. Si consideramos tres personas, se necesitarían 2 neuronas en la capa de salida para poder clasificarlas, donde el conjunto de las neuronas en la capa de salida representará en código binario el número decimal asociado con la persona. La persona 2 la representaríamos con 2 números: {1,0}, donde proponemos los valores 0 para la primer neurona en la capa de salida y 1 para la segunda neurona. Así podemos clasificar la información en un total de 2^s clases, donde s es el número de neuronas en la capa de salida. Con esta dinámica para el problema en donde usamos la información de la voz de 6 personas, bastan 3 neuronas en la capa de salida para poder clasificar los datos introducidos y seguiremos llamando a la primer persona, la persona 1 (no consideraremos la persona 0). De aquí en adelante llamaremos a este método "método binario". Si seleccionamos por ejemplo los datos pertenecientes a una palabra hablada por la persona 3 equivaldría a escoger los objetivos como

$$3 \rightarrow \{0, 1, 1\} \rightarrow \{t_1 = 0.1, t_2 = 0.9, t_3 = 0.9\}$$

Se usaron los parámetros dentro de la red que se habían tomado en cuenta en los entrenamientos anteriores (función de activación tangente hiperbólica, 15 neuronas en la capa oculta, $\gamma = .05$, $\theta = 0$, $\mu = 0$, pesos aleatorios iniciales entre $[-1, 1]$), con la normalización de la Ecuación 4.5 y se consideró primero solo una palabra, para incrementar posteriormente una por una. Para la palabra "hola" se usaron 7000 iteraciones para el entrenamiento de la red. Para las palabras "hola" y "bienvenido" se utilizaron 20000 iteraciones. Al agregar adicionalmente la palabra "adiós" a las anteriores, se necesitaron 40000 iteraciones para el entrenamiento. Finalmente se introdujo la palabra "felicidades" pala el entrenamiento simultaneo de las 4 palabras y se utilizaron 120000 iteraciones, pero por desgracia no obtuvimos los resultados de los objetivos propuestos para la persona 4, aún cuando se intentaron variando parámetros como la cantidad de neuronas y la constante de aprendizaje. Donde no se obtuvo el resultado esperado, fue en la misma palabra y con la misma persona donde antes vimos que se alejaba mucho de los objetivos cuando se entrenó individualmente (ver Tabla 4.19). Así que se optó por cambiar un poco la normalización para los datos perteneciente a esta

palabra para las seis personas, utilizando la siqguiente expresión:

$$\xi_i = \left(\nu_i - \sum_{J=1}^{10} \frac{\nu_j}{10}\right) \frac{1}{800} \tag{4.7}$$

Volviendo a introducir las 4 palabras para el entrenamiento simultaneo y usando los parámetros anteriores a exepción de esta nueva normalización para la palabra "felicidades", el resultado para el error de un total de 100000 iteraciones lo podemos ver en la Figura 4.20.

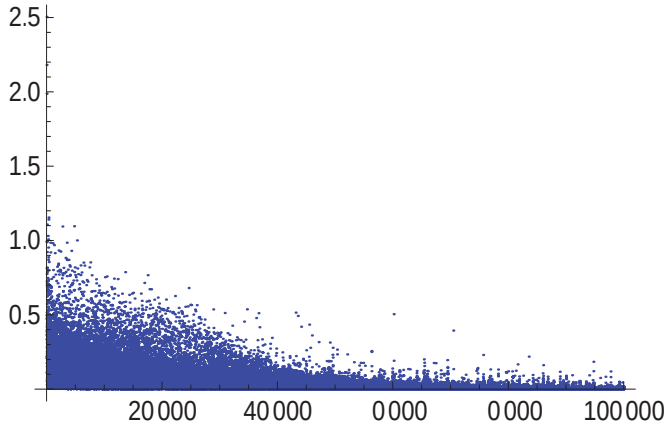

Figura 4.20: Disminución del error $E = \frac{1}{2}\sum_{k=1}^{salida}(t_k - s_k)^2$ para 100000 iteraciones en el entrenamiento simultaneo de la red para las palabras "hola", "bienvenido", "adiós" y "felicidades" y utilizando el método binario. Parámetros: Función tangente hiperbólica, 16 neuronas en la capa oculta, $\gamma = .05$, $\theta = 0$, $\mu = 0$, pesos aleatorios iniciales entre [-1,1] y valores objetivo de $\{.9, .1\}$.

En la Tabla 4.20 se observa el tiempo de ejecución del entrenamiento tanto para el primer método empleado para la selección de los objetivos propuestos, como para el método binario empleado después. Además de necesitar un menor número de neuronas de salida, también se necesitó un número menor de iteraciones para el entrenamiento con el método binario.

Los resultados obtenidos en las neuronas de salida después de propagar la información por la red con los pesos obtenidos del entrenamiento para las cuatro palabras con el método binario, se encuentran en la Tabla 4.21.

	Tiempo (segundos)			
Método	1 Palabra	2 Palabras	3 Palabras	4 Palabras
1er método	665	1656	3388	9136
Binario	271	836	1593	3958

Tabla 4.20: Tiempo de ejecución del entrenamiento de la red para ambos métodos. En la mayoría de los entrenamientos el método binario requiere menos de la mitad de tiempo en comparación con el primer método.

Neurona	Persona 1	Persona 2	Persona 3	Persona 4	Persona 5	Persona 6
Palabra "hola"						
s_1	0.1681	0.1040	0.1647	0.8940	0.8923	0.9107
s_2	0.0331	0.9843	0.8440	-0.0473	0.0182	0.8488
s_3	0.9325	0.1211	0.8992	0.0419	0.9219	0.0457
Palabra "bienvenido"						
s_1	0.1286	0.1374	0.1132	0.9178	0.9088	0.8860
s_2	0.0461	0.9905	0.8749	0.0798	0.0296	0.8605
s_3	0.9767	0.0714	0.8917	0.0951	0.8995	0.1580
Palabra "adiós"						
s_1	0.1825	0.1522	0.1667	0.8919	0.8854	0.9966
s_2	0.0245	0.8877	0.8489	0.0665	0.0436	0.8868
s_3	0.9173	0.0861	0.8305	0.0740	0.8697	0.0994
Palabra "felicidades"						
s_1	0.1715	0.1256	0.1809	0.8931	0.9563	0.8870
s_2	0.0223	0.8789	0.8237	0.0453	0.0427	0.9197
s_3	0.8488	0.1016	0.8150	0.1133	0.9215	0.1020

Tabla 4.21: Valores de las neuronas de salida al introducir la información de las 4 palabras para las 6 personas después del entrenamiento de la red utilizando el método binario. Los valores aproximados a 0.1 representan ceros mientras los valores próximos a 0.9 representan unos. Los 3 valores en las neuronas de salida en cada columna representan en código binario el número decimal de la persona que dijo la palabra.

4.3.1 Extrapolación de los resultados

Anteriormente hemos hablado de la capacidad que tienen las RNA para extrapolar los resultados al introducir conjuntos de datos diferentes a los que se usaron durante el entrenamiento de la red. Así que ahora veremos que tanto margen de error pueden tener estos datos de entrada, con respecto a los datos que utilizamos para el entrenamiento. Lo que se hizo, fue tomar los vectores que teníamos originalmente de cada palabra empleada en el entrenamiento de la red y a cada una de las entradas de los vectores que contienen las frecuencias normalizadas, se les aumentó el uno por ciento del valor de las mismas. Para dejarlo más claro, tomaremos como ejemplo los valores de las frecuencias normalizadas de la palabra "hola", perteneciente a la persona 1 (ver Tabla 4.8):

$$\{\ \xi_1 + \tfrac{1}{100}|\xi_1|\ ,\ \xi_2 + \tfrac{1}{100}|\xi_2|\ ,\ \ldots,\ \xi_{10} + \tfrac{1}{100}|\xi_{10}|\ \}$$

$$\Downarrow$$

$$\{\ \text{-1.08+0.018}\ ,\ \text{-0.83+0.0083}\ ,\ \ldots,\ \text{1.54+0.0154}\}$$

Modificando de esta manera cada una de las entradas de todas las palabras para las seis personas y utilizando los pesos que dieron como resultado los valores de la Tabla 4.21, resultó que la red clasificó satisfactoriamente los datos, es decir, identificó a que persona pertenecía la voz. Usando el procedimiento anterior pero en vez de sumar, se restó el uno por ciento a las entradas, se obtuvo que la red clasificó la información adecuadamente. Al modificar así la información estamos introduciendo ruido artificial, es decir, estamos alterando la señal que tenemos originalmente, para tratar de simular como una primera aproximación el ruido que se generaría en la señal al cambiar las condiciones al momento de cuando se grabaron las palabras. Se siguió aumentando el porcentaje con el que se alteró la señal y hasta un 6 por ciento de ruido la red trabajó correctamente. Al tener un margen de ruido de 7% la red ya no clasificó adecuadamente la información. Con esto nos percatamos de que la RNA acepta un margen de ruido de hasta ±6%. Esto ocurre si introducimos el mismo margen de ruido en cada una de las entradas, pero... ¿Cambia el resultado si introducimos diferentes porcentajes de ruido en cada una de las entradas de las palabras?

La respuesta es que si cambia. Tomando como cota este ±6%, se prosigió a tomar diferentes margenes de ruido aleatoriamente entre $[-\frac{6}{100}, \frac{6}{100}]$ para cada una de las entradas

de las 4 palabras y lo que encontramos es que al modificar así las entradas y usar los pesos encontrados con el entrenamiento, ya no pudo clasificar la red adecuadamente la información. De esta forma la red realiza el reconocimiento satisfactoriamente, usando sólo el 1% de ruido. Para llegar a esta respuesta se pusieron aleatoriamente estos margenes de ruido y se verificó si existía al menos una palabra que no se clasificara correctamente, realizando 20 corridas para cada porcentaje de ruido y consiguiendo los resultados mostrados en la Tabla 4.22.

Margen de ruido	Clasificación incorrecta
±1%	0
±2%	1
±3%	3
±4%	5
±5%	12
±6%	19

Tabla 4.22: La primer columna muestra el intervalo donde se escogió aleatoriamente el ruido que se introdujo en cada una de las entradas de los vectores pertenecientes a las palabras utilizadas previamente en el entrenamiento. La segunda columna muestra las corridas en las que al menos una palabra no se clasificó correctamente de acuerdo a la persona a la que pertenecía, para un total de 20 corridas.

Por desgracia este tipo de resultados en donde solo algunas entradas del vector que introduciremos en la RNA tienen ruido, es más común al momento de hacer grabaciones de sonido. Al usar un micrófono de menor calidad y grabar las palabras con las que se entrenó la red, las frecuencias bajas de las palabras se ven afectadas de mayor manera y la red no fue capaz de realizar la clasificación utilizando los pesos que encontramos aquí.

Para lograr este propósito sería necesario entrenar la red, utilizando grabaciones mediante el uso de diferentes micrófonos para ampliar el alcance de la extrapolación. También se optó en usar todo el rango de frecuencias audible pensando a futuro, con el propósito de clasificar o identificar otro tipo de sonidos y no únicamente voces. Sería una mejor aproximación realizar el estudio para un rango de frecuencias en donde se

encuentran (en promedio) las voces de las personas. Por último tal vez se necesite usar alguna otra herramienta matemática para seleccionar de mejor manera las frecuencias que caracterizan las palabras.

Hasta este punto, hemos logrado los objetivo principales de este proyecto, los cuales consistían en primer lugar, obtener información de como modificar los parámetros dentro de una RNA para poder minimizar el error entre los resultados que se esperan obtener y lo que resulta de la propagación de la información por la red y en segundo lugar, logramos que la red neuronal relacione las palabras guardadas, con las personas a las que pertenece la voz de tales palabras. Se podría dotar a la red con capacidad de hacer mas clasificaciones, como identificar el sexo de la persona a la que pertenece la voz de cada una de las cuatro palabras que se analizaron y no solo con una palabra como se hizo anteriormente, pero se tendrían que hacer normalizaciones diferentes para cada palabra y ademas, tal vez, una selección de frecuencias diferentes a las que se han usado aquí. Debido a las limitaciones en cuanto al tiempo para desarrollar estas tareas adicionales, se dará por cuncluído este proyecto de tesis.
Se tiene pensado continuar el estudio con los conocimientos adquiridos en este proyecto de tesis para continuar el análisis de señales de audio mas complejas y mejorar lo que se hizo hasta este momento.

Finalmente expondremos las conclusiones a las que hemos llegado con el estudio de las redes neuronales artificiales como herramienta para la solución de problemas.

CAPÍTULO 5

CONCLUSIONES

En este proyecto de investigación nos pudimos dar cuenta de la flexibilidad que tienen las redes neuronales artificiales para resolver diferentes tipos de problemas, debido a la capacidad que tienen para resolverlos mediante la generalización de la solución en base a un entrenamiento previo, tomando como datos iniciales los valores que tienen las variables que caracterizan al problema, en nuestro caso, se utilizó el espectro de frecuencias correspondientes a cada palabra grabada, para su clasificación de a cuerdo a la persona a la que pertenecía la voz. Como los datos que se introducen dentro de la red pueden representar cualquier tipo de variables dependiendo del sistema físico que se quiere resolver (o cualquier otro problema que se pueda plantear para ser solucionado utilizando una RNA backpropagation), podemos predecir el comportamiento de la red o tener una mejor idea de los cambios que se pueden realizar en los parámetros de la misma, para llegar a tener mejores resultados en el entrenamiento, con el propósito de obtener los objetivos propuestos en las neuronas de salida al usar las RNA aquí presentadas.

Después de analizar los diferentes parámetros que constituyen una RNA tipo backpropagation y como afectan a la misma durante el entrenamiento, lo principales aspectos que se tienen que tomar en consideración al implementar tales redes son los siguientes:

- Seleccionar un conjunto de datos iniciales adecuados para introducirlos dentro de la red, que tengan información esencial del problema (variables que definan el sistema) que se va a resolver.

- Escoger una función de activación y de ser necesario, adaptar la información que se va a introducir para que se encuentre dentro del intervalo de sensibilidad de tal función.

- Proponer valores de los objetivos, de manera que sea posible que la función de activación que se escogió tome esos valores.

- Es importante tener en cuenta el valor o el intervalo de valores en los que se pueden inicializar los pesos para el entrenamiento de la red.

- Si no tenemos una idea clara de donde se pueden introducir los términos bias (θ) y del valor de los mismos, es mejor no tomarlos en cuenta como primera aproximación ($= 0$).

- Tratar de tener el menor número posible de neuronas en la capa de salida para disminuir el tiempo de ejecución al entrenar la red.

- Si al momento de entrenar la red el error converge a cero, pero requiere de muchas iteraciones debido a que disminuye muy lentamente, se puede incrementar el número de neuronas en la capa oculta para que requiera de menor número de iteraciones. También puede ser que se necesite introducir de otra manera la información (otra normalización de los datos).

- Si el error oscila demasiado, disminuir el valor del parámetro de aprendizaje (γ) para evitar estas oscilaciones. En caso de que el error continúe oscilando después de modificar γ, introducir la constante del momento (μ).

- Si una vez que se a hecho todo lo anterior (considerando que se han realizado las pruebas suficientes), seguimos sin obtener buenos resultados al entrenar la red, se necesita realizar nuevamente el procedimiento anterior probando con otros valores de los objetivos o de la función de activación, antes de pensar en introducir otro tipo de parámetros o incluso otro tipo de estructura.

- Si a los datos con los que se entrenó la red, se les agrega ruido en proporciones iguales a cada uno de estos datos que se introducirán en la misma, el margen de ruido que permite la red para trabajar adecuadamente, es mayor a cuando se les ponen proporciones diferentes de ruido a cada uno de ellos.

Tomando en consideración los puntos anteriores desde un inicio, tomaremos mejores decisiones al momento de resolver problemas que involucren otro tipo de variables, en cuanto a los parámetros de la red que hay que modificar para obtener resultados satisfactorios al entrenar la red. Con esto reduciremos el tiempo de implementación de los problemas de mayor complejidad que resolveremos mas adelante.

5.1 Trabajo futuro

Como continuación a este proyecto de investigación, en el futuro se desea optimizar la red mediante la paralelización del código, ya que la estructura de las RNA es ideal para este propósito. Para paralelizar nuestro programa es necesario hacer uso de tarjetas gráficas (GPUs) o de clusters de computadoras que cuenten con un gran número de procesadores, para que sea posible realizar varios cálculos simultaneamente en cada uno de los procesadores, por lo que el tiempo total de procesamiento se ve reducido. Esto nos permite que podamos introducir una mayor cantidad de información para procesar, pudiendo resolver así problemas más complejos que requieran hacer un mayor número de cálculos.

También se desea implementar otro tipo de algoritmos para que trabajen junto con las RNA, tal es el caso de los algoritmos genéticos, los cuales pueden ser utilizados para encontrar una estructura óptima de la red, como lo son el número de neuronas necesarias en la capa oculta para lograr un mejor desempeño. Con esto disminuimos el número de parámetros libres dentro de la red, ya que por ejemplo, nosotros tuvimos que estar seleccionando el número de neuronas en la capa oculta de la red, en base a prueba y error.

Dentro del análisis de señales digitales de audio, se desea emplear más herramientas matemáticas que nos sean útiles para hacer una mejor selección de los datos que se introducirán en la red, esto con el fin de estudiar sonidos más complejos o conjunto de sonidos como se tienen en la música. Una aplicación podría ser la selección de frecuencias correspondientes a algún instrumento en particular, del conjunto total de instrumentos presentes en alguna canción.

Por último se quiere hacer uso de las RNA en problemas que involucren sistemas físicos más complicados. Dentro de los temas de mi interes se encuentran aquellos donde se emplean propiedades cuánticas de la materia para aplicarlas junto con el uso de RNA, para crear nuevas redes o para analizar propiedades de materiales en los que los efectos de la mecánica cuántica están presentes.

Esperamos que con este proyecto de investigación hayamos podido transmitir al lector el alcance que pueden tener las RNA para la resolución de problemas, dejando claro que sólo empleamos una estructura particular de red (Backpropagation), ahora imagine todo el tipo de problemas que pueden ser resueltos al existir decenas de estructuras de RNA.

BIBLIOGRAFÍA

[Chapra and Canale, 2007] Chapra, S. C. and Canale, R. P. (2007). *Métodos Numéricos para ingenieros*. Mc Graw Hill, 5 edition.

[Dymarski, 2001] Dymarski, P. (2001). *Hidden markov models, theory and applications*. InTech.

[Feldmand and Ballard, 1982] Feldmand, J. and Ballard, D. (1982). Connectionist models and their properties. *Cognitive Science*.

[Freeman and Skapura, 1991] Freeman, J. A. and Skapura, D. M. (1991). *Neural networks: algorithms, applications and programming techniques*. Addison-Wesley Publishing Company.

[Graupe, 2007] Graupe, D. (2007). *Principles of Artificial Neural Networks*. World Scientific Publishing, 5 edition.

[Hayes, 1999] Hayes, M. H. (1999). *Schaum's Outline Of Digital Signal Processing*. McGraw-Hill, 1 edition.

[Kohonen, 1988] Kohonen, T. (1988). *Self-Organizing Feature Maps*. Springer Berlin Heidelberg.

[Kroese and Van der Smagt, 1996] Kroese, B. and Van der Smagt, P. (1996). *An Introduction to neural networks*. The University of Amsterdam, 8 edition.

[Lagaris et al., 1997a] Lagaris, I., Likas, A., and Fotiadis, D. (1997a). Artificial neural network methods in quantum mechanics. *University of Ioannina*.

[Lagaris et al., 1997b] Lagaris, I., Likas, A., and Fotiadis, D. (1997b). Artificial neural networks for solving ordinary and partial differential equations. *University of Ioannina*.

[Orfanidis, 2010] Orfanidis, S. J. (2010). *Introduction To Signal Processing*. Rutgers University.

[Popa, 2012] Popa, R. (2012). *Genetic Algorithms in Applications*. InTech.

[Proakis and Manolakis, 2006] Proakis, J. G. and Manolakis, D. G. (2006). *Digital Signal Processing*. Prentice Hall, 4 edition.

[Rojas, 1996] Rojas, R. (1996). *Neural networks. A systematic introduction*. Springer.

[Rosenblatt, 1958] Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*.

Resumen

Para resolver problemas físicos es necesario recurrir al uso de métodos numéricos, dado que a veces las ecuaciones que hay que resolver no se pueden solucionar por métodos analíticos o son demasiado complicadas y es más fácil resolverlas numéricamente. Uno de los métodos numéricos que se está usando actualmente debido a que se pueden implementar para resolver problemas en prácticamente cualquier área de las ciencias naturales, es el uso de redes neuronales artificiales (RNA). Este tipo de algoritmos basan su funcionamiento en la forma en que se transmite la información en las redes neuronales biológicas que tenemos en nuestro cerebro. Algunas de las aplicaciones que tienen las RNA, es que, pueden ser usadas tanto para resolver ecuaciones diferenciales ordinarias, como para ecuaciones diferenciales parciales, así como también para problemas de valor en la frontera que suelen describir sistemas físicos.
En particular nosotros usamos las RNA y la transformada discreta de Fourier (TDF) para análisis de señales digitales de audio, con el fin de utilizar grabaciones de palabras de un grupo de personasseleccionado y nuestro algoritmo se encarga de clasificar las palabras de manera que identificamos que persona dijo la palabra, es decir, realizamos un reconocimiento de voz.

Palabras clave:
Redes Neuronales Artificiales
Transformada Discreta de Fourier
Reconocimiento de voz

Abstract
To solve physical problems is necessary to use numerical methods, since sometimes the equations to be solved can't be solved by analytical methods or are too complicated and it's easier to solve numerically. One of the numerical methods currently being used because they can be implemented to solve problems practically in any area of natural sciences, is the use of artificial neural networks (ANN). Such algorithms base their operation on the way the information in our brain is transmitted. Some of the applications that have the ANN, is that can be used to solve ordinary differential equations, partial differential equations as well as for problems of boundary value which often describe physical systems.
In particular we use the ANN and the Discrete Fourier Transform (DFT) for analysis of digital audio signals in order to use recordings of words of a selected group of people and our algorithm is responsible for sorting the words so we can identify the person that said the word, that is to say, we perform speech recognition.

Keywords:
Artificial Neural Network
Discrete Fourier Transform
Speech recognition.