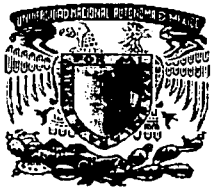
18



UNIVERSIDAD NACIONAL AUTÓNOMA
DE MÉXICO

FACULTAD DE INGENIERÍA

SISTEMA DE RECONOCIMIENTO DE COMANDOS DE VOZ, MEDIANTE REDES NEURONALES ARTIFICIALES

T E S I S

QUE PARA OBTENER EL TÍTULO DE

INGENIERO MECÁNICO ELECTRICISTA

P R E S E N T A :

P E D R O C A M P O S V A R G A S

DIRECTOR: ING. JUAN MANUEL GÓMEZ GONZÁLEZ

MÉXICO, D. F. 2002

DEDICADA A :

Mis Padres.

Con mucho cariño y respeto les dedico esta tesis como un reconocimiento a las innumerables cosas que han hecho por mi, por sus sabios consejos, por apoyarme con paciencia en todos mis proyectos y quienes gracias a su trabajo me han permitido alcanzar todas mis metas.

Martha Alvarado M.

Quien ocupa un lugar muy especial en mi corazón, con quien he compartido muchos momentos memorables en mi vida y quien me ha motivado con su cariño y dedicación a ser una mejor persona. A ella con muchísimo cariño.

Mis hermanas Ale, Rosita, Sarita y Tina.

Quienes me han visto crecer, han dedicado un poco de tiempo para darme algún consejo y se han preocupado por mi en todo momento.

AGRADECIMIENTOS.

A la Universidad Nacional Autónoma de México :

Por abrirme sus puertas y mostrarme nuevos horizontes.

A mi director de tesis Ing. Juan Manuel Gómez G. :

Por su paciencia, confianza y consejos para la realización de esta tesis, pero principalmente por su amistad.

Al Departamento de Control de la Facultad de Ingeniería :

Por permitirme hacer uso de sus instalaciones.

Al M. I. Daniel Kornhauser Eisenberg :

Por sus valiosos conocimientos en el tema de reconocimiento de voz.

Al Ing. Cesar Carvajal :

Por las facilidades para utilizar el laboratorio de cómputo.

A mis cuñados Jesús y Luis :

Por sus palabras de aliento, su consejos y por apoyarme en todos los momentos en los que los he necesitado.

A todos mis tíos y primos:

Por darme ánimos y apoyarme con sus consejos en las diversas etapas de mi vida.

A mis amigos Fabiola e Israel :

Por todos estos años en los que hemos compartido penas y alegrías.

INDICE

INTRODUCCIÓN

A lo largo de la historia el hombre ha buscado técnicas e instrumentos que le permitan realizar sus tareas cotidianas fácilmente y en el menor tiempo posible. Es así como el ritmo acelerado en el que vivimos hoy en día crea la necesidad de fabricar máquinas que realicen sus tareas de forma rápida y sencilla, pero además que el medio de comunicación por el que reciban instrucciones sea más cómodo que el hecho de apretar un botón o utilizar un teclado, esto es posible mediante el empleo de comandos de voz, tecnología que ha sido tomada por muchos como un tema de ciencia ficción, pero que gracias a las técnicas de procesamiento digital de señales se postula como una de las más prometedoras del futuro.

El deseo de los científicos por construir máquinas que se comuniquen con el hombre por medio de palabras y que además sean capaces de interpretar la voz, más que un simple capricho por igualar las capacidades humanas se debe al hecho de que la forma más simple de comunicación entre los seres humanos es el habla. El hombre aprende a hablar mucho antes de saber leer o escribir, además es más rápido pronunciar una palabra que escribirla y cuando se establece una comunicación a distancia, como es el caso de una conversación telefónica no es necesario emplear las manos o tener la vista fija sobre nuestro interlocutor para que la transmisión de mensajes sea exitosa.

El reconocimiento artificial de voz se define como:

La habilidad de un sistema para reconocer e interpretar la voz humana con el fin de realizar una acción.

La clasificación de un sistema de reconocimiento de voz (SRV) se basa en la forma en la que el usuario pronuncia las palabras y en el número de locutores que pueden ser reconocidos por la aplicación.

Tomando en cuenta la forma en la que el usuario pronuncia las palabras a ser reconocidas por el SRV los sistemas se dividen en:

- Reconocimiento no continuo del habla
- Reconocimiento continuo del habla.

Los sistemas que reconocen el habla de forma no continua son fáciles de producir, pero difíciles de usar debido a que requieren que el locutor pronuncie pausadamente las palabras de una frase. Se considera que este tipo de sistemas no proporcionan una interfaz natural, sin embargo permiten reconocer un mayor número de patrones de voz.

Los sistemas de reconocimiento continuo del habla son más difíciles de producir, pero más fáciles de usar. En este tipo de aplicaciones los usuarios emplean su ritmo natural de pronunciación sin necesidad de cortar las palabras de una frase o de esperar alguna señal

que les indique que pueden continuar hablando. En los sistemas de reconocimiento de voz tanto continuo como no continuo se requiere que el locutor pronuncie las palabras lo más claro posible, además de ser sensibles al ruido ambiental excesivo.

En cuanto a la clasificación de los sistemas de reconocimiento de voz por el número de usuarios, la división es la siguiente: Sistemas dependientes del usuario, sistemas múlti-usuario y sistemas independientes del usuario.

En el primer tipo, la base de datos compuesta por las palabras que se han de reconocer se forma con la voz de un solo usuario. En los sistemas múlti-usuario, la base de datos se forma con la voz de varios usuarios pero reconoce exclusivamente las voces contenidas en dicha base. Los sistemas independientes del usuario son similares a los sistemas múlti-usuario, pero con la diferencia de que son capaces de reconocer no solo las voces contenidas en la base, sino también las de otras personas.

En las aplicaciones de reconocimiento de voz orientadas a un solo usuario, como algunos programas de dictado por computadora, el objetivo principal es tener una base de datos que contenga la mayor cantidad de palabras posible, incluyendo las diferencias que existen debido a la acentuación.

En los sistemas múlti-usuario, como es el caso de aplicaciones para telefonía, el objetivo principal es encontrar un método que permita generalizar las palabras contenidas en su base para cualquier persona. Los sistemas de este tipo tienen una base de datos reducida debido a que resulta complejo manejar simultáneamente las variaciones en la pronunciación de las palabras y palabras fonéticamente parecidas.

A medida que el programa tiene que generalizar es necesario acotar el problema, esto se puede hacer pidiéndole al locutor que pronuncie palabras pausadas, reduciendo lo más que se pueda la influencia del ruido ambiental, teniendo un solo locutor o limitando el número de palabras.

El volumen de información requerido para reconocer la voz de diferentes personas es grande y aumenta a medida que incluimos variaciones en la pronunciación. Por ejemplo el acento que varia de una región geográfica a otra o los cambios que se presentan en la voz cuando cambia el estado de ánimo de una persona. Por lo tanto es necesario entonces que un sistema de reconocimiento de voz incluya métodos que le permitan agrupar de forma sencilla el mayor número de variaciones de voz posible.

Una herramienta cuya utilidad se ha comprobado no solo en la generación e interpretación de lenguaje sino también en el reconocimiento de patrones complejos y el procesamiento de imágenes son las Redes Neuronales Artificiales (RNA).

Las Redes Neuronales Artificiales permiten obtener por medio de algoritmos computacionales, soluciones a problemas complejos que requieran un cierto grado de aprendizaje.

Una RNA es un sistema no lineal integrado por elementos llamados nodos los cuales se conectan entre si formando una estructura de red. Estos nodos tratan de imitar la forma en la que trabajan las neuronas.

Una de las formas en la que una red adquiere "conocimiento" es por medio de ensayo y error. A este tipo de aprendizaje se le conoce como entrenamiento supervisado. Una explicación sencilla de los pasos que se siguen en el entrenamiento de una RNA es la siguiente: se muestran a la red los patrones que deben ser aprendidos, dichos patrones son multiplicados por determinados valores llamados pesos. Los valores que resultan son operados matemáticamente para generar una salida. Esta salida se compara con la salida deseada por medio de un error que es la diferencia entre ambas salidas; si el error es menor a un valor previamente establecido entonces el entrenamiento termina. En caso contrario, el sistema realiza correcciones en los pesos y continúa generando salidas hasta alcanzar el error deseado.

En el entrenamiento de una Red Neuronal Artificial se debe "enseñar" a la red la mayor cantidad posible de patrones de una misma clase, además se deben presentar los datos que contienen la información más representativa con el fin de que el sistema sea capaz de clasificar cada patrón correctamente. En el caso de los SRV los patrones estarían constituidos por datos provenientes del procesamiento de las señales de voz.

Una Red Neuronal correctamente entrenada debe poseer una buena capacidad de generalización, lo que significa que ha "aprendido" de sus patrones de entrenamiento y que responderá adecuadamente ante patrones que no se hayan visto con anterioridad.

Aún cuando las RNA permiten clasificar una gran cantidad de señales de voz, no se ha podido crear un sistema de reconocimiento de voz 100 % eficaz debido a que la voz es un fenómeno difícil de modelar. La complejidad para obtener un algoritmo que sea capaz de reconocer la voz de diferentes usuarios aumenta cuando se toman en cuenta las frecuencias altas que componen la voz de niños y mujeres. Se requiere entonces que los métodos empleados para el procesamiento de voz sean capaces de obtener la mayor cantidad de información posible con pocos datos. La obtención de los datos más representativos de las señales permite eliminar información redundante, provocando que la red neuronal pueda realizar una correcta clasificación tanto de los patrones de entrenamiento como de prueba.

Las principales aplicaciones del reconocimiento de voz se han dado en telefonía, sin embargo el área de aplicación es extensa y variada, desde el simple dictado a una computadora hasta circunstancias en las que se tengan ocupadas las manos y la vista o mejor aún para ayudar a discapacitados que no pueden usar teclados o botones.

Dentro de las aplicaciones disponibles actualmente en el mercado se encuentran las siguientes:

Programas de dictado

Dentro de los cuales se encuentran Dragon NaturallySpeaking de la empresa Dragon Systems y ViaVoice de la empresa IBM. Ambos productos permiten la captura de texto en computadoras personales así como la ejecución de algunos comandos por medio de la voz.

Programas de telefonía.

Entre estos programas se encuentra Lucent Speech Server diseñado y distribuido por la empresa Lucent Technologies (Bell Labs).

La aplicación de este conmutador telefónico permite la marcación por medio de la voz. Esta función permite a los usuarios hacer llamadas simplemente pronunciado el número telefónico deseado o diciendo el nombre de la persona a la que se desea llamar.

Otro ejemplo es SMARTRecognizer (SpeechWorks). Este conmutador puede reconocer frases y oraciones completas sin necesidad de entrenamiento. además de soportar reconocimiento continuo del habla.

Los primeros sistemas para reconocimiento del habla fueron construidos empleando ingeniosos circuitos analógicos para detectar las características de la voz y una lógica simple que realizaba la clasificación de diferentes palabras. Sin embargo estos artefactos eran difíciles de producir, requerían de una pronunciación casi perfecta y además eran costosos. Estas desventajas limitaban el uso de los SRV a centros de investigación o grandes empresas.

Debido a que la implementación de un sistema de reconocimiento de voz es relativamente económica empleando procesamiento digital de señales y gracias al surgimiento de computadoras digitales cada vez más veloces y baratas, las aplicaciones de reconocimiento desarrolladas con fines comerciales o de investigación se encuentran cada día al alcance de más y más personas alrededor del mundo.

En la presente tesis se explican los conceptos básicos empleados en la construcción de sistemas de reconocimiento de voz así como las principales características de las Redes Neuronales Artificiales.

El objetivo es sentar las bases para el desarrollo de un sistema de reconocimiento no continúo del habla e independiente del usuario, capaz de reconocer algunos comandos. Para ello se realizaron prueba con diferentes arquitecturas de redes neuronales, empleando como método de aprendizaje el algoritmo de retropropagación del error. Para la obtención de las características de la voz se probaron dos métodos: los coeficientes de predicción lineal y los coeficientes ceptrum en escala mel, con la finalidad de determinar con cual de ellos obteníamos un mejor desempeño.

Se optó por encontrar primero un modelo con vocales debido a que su número de muestras es pequeño y nos permite entrenar la red más rápidamente, una vez obtenido el modelo para vocales se hicieron algunos ajustes para lograr que este funcionara con palabras aisladas.

CAPITULO I

SISTEMAS DE RECONOCIMIENTO DE VOZ

SISTEMAS DE RECONOCIMIENTO DE VOZ

1.1 Primeros Sistemas de Reconocimiento de Voz

La primera máquina "capaz" de reconocer señales de voz fue un perro de juguete llamado Radio Rex, el cual era vendido en la década de 1920. Este tenía un dispositivo que respondía a sonidos con frecuencias cercanas a la frecuencia fundamental de la vocal "e" en la palabra Rex. Cuando un sonido contenía esta frecuencia y además su potencia rebasaba cierto umbral el juguete caminaba y daba la sensación de responder a su nombre.

A finales de 1940 y principios de 1950 los laboratorios Bell construyeron un sistema capaz de reconocer diez dígitos pronunciados por un solo usuario. Este sistema realizaba una comparación de las palabras contenidas en su base con las señales que recibía del exterior, si la señal entrante se parecía con un cierto grado de error a alguna de las señales almacenadas este la tomaba como valida e indicaba que dígito se había pronunciado.

En 1959 Fry y Denes construyeron en la universidad de Coller, Inglaterra un instrumento que reconocía cuatro vocales y nueve consonantes utilizando un método similar al empleado por el sistema construido por los laboratorios Bell.

A finales de 1960 y principios de 1970 se mejoraron los sistemas de reconocimiento de voz debido, principalmente al surgimiento de algoritmos que permitieron extraer con menos datos las características de la voz, dentro de estos métodos se encuentran la transformada rápida de Fourier o FFT, la aplicación de los coeficientes de predicción lineal a la voz y la implementación del análisis cepstral en problemas relacionados con la voz. Estos dos últimos métodos se explicaran con detalle en el capitulo III.

La industria bélica también contribuyó al desarrollo de los sistemas de reconocimiento de voz pues vislumbró en ellos un instrumento con muchas aplicaciones. La participación de los militares fue indirecta, es decir a través de apoyos económicos a investigaciones en este campo. Bajo este patrocinio se construyeron varios sistemas que fueron capaces de reconocer un número grande de palabras (aproximadamente 60,000), además se crearon los primeros sistemas que reconocían palabras sin importar que usuario las pronunciara. [8]

Como se puede apreciar el desarrollo y utilización de los SRV estuvo limitado a grandes empresas, la industria militar y las universidades, pero gracias al avance tecnológico en nuestros días y a la popularidad y bajo costo de las computadoras, el desarrollo de sistemas de reconocimiento de voz se ha diversificado permitiendo que estos se encuentren prácticamente al alcance de cualquier persona, a través aplicaciones comerciales o de distribución libre a través de la red.

1.2 Aplicaciones de los Sistemas de Reconocimiento de Voz.

Las interfases que utilizan a la voz como una forma de interacción hombre-máquina han sido fácilmente aceptados por la gente común debido a que presentan claras ventajas sobre las interfases tradicionales como el teclado o el ratón. Algunas ventajas de utilizar la voz humana son:

Alta velocidad de entrada: algunos experimentos han revelado que la información se puede capturar tres veces más rápido dictándola que escribiéndola a través de un teclado. [7]

No se necesita entrenamiento: La generación de la voz es una acción muy natural en los humanos y no necesitamos un entrenamiento previo para poder interactuar con el sistema, esto facilita que un mayor número de personas incluyendo aquellas que nunca han utilizado una máquina de escribir, una computadora o incluso que no sepan leer o escribir puedan aprovechar esta tecnología.

Procesos en paralelo con otras acciones: Se pueden realizar otras tareas con las manos mientras le damos instrucciones a una computadora.

Las ventajas anteriores han sido aprovechadas en diversas aplicaciones dentro de las que se encuentran:

Las telefónicas: En este campo los sistemas de reconocimiento de voz han sido empleados principalmente para que el usuario proporcione respuestas a peticiones como el número telefónico que se desea marcar o el número de extensión, sin la necesidad de apretar ningún botón.

Operaciones con manos libres: Existen muchas situaciones en las cuales una persona tiene ocupadas sus manos y no puede utilizarlas para controlar algún aparato. Un ejemplo es el control del lente de un microscopio durante una intervención quirúrgica en la cual el médico no puede distraer su atención ni sus manos en ajustar la imagen captada por el microscopio.

Discapacitados: El reconocimiento de voz es una interfaz natural y muy útil para personas con inmovilidad de brazos o que hayan perdido el sentido de la vista. Los sistemas de reconocimiento de voz les permiten a las personas discapacitadas controlar por medio de computadoras algunos aparatos electrónicos empleados en el hogar o por ejemplo abrir remotamente la puerta de su casa.

Los sistemas de reconocimiento de voz son útiles no solo para las personas con alguna discapacidad, sino también para todos en general, porque como ya se dijo son una interfaz más natural que el teclado o el ratón.

Dictado: Es quizá la aplicación más comercial de nuestros días y la más conocida por el público en general. El objetivo de está aplicación es disminuir el tiempo en el cual

una persona captura texto en una computadora y además les da a sus usuarios la libertad de realizar otras actividades al mismo tiempo que se encuentran dictando.

Traducción: Una aplicación con mucho futuro es la traducción de palabras o frases dichas en un idioma a otro. Un ejemplo de este tipo de aplicaciones es el proyecto Verbmobil desarrollado en Alemania el cual facilitará las conversaciones entre alemanes y japoneses

1.3 Problemas que se presentan en el desarrollo de un Sistemas de Reconocimiento de Voz.

La forma más fácil de comunicación entre los seres humanos es el habla, esta habilidad del hombre para comunicarse es tan natural, que gran parte de las personas que no tienen una educación técnica o científica se preguntan por que no existen instrumentos que sean capaces de interpretar la voz humana con rapidez y sencillez. El análisis de la voz es una tarea complicada debido a que la voz presenta cambios significativos no solo de una persona a otra, sino también cambios en una misma persona. La variación de la voz provoca que el desarrollo de un sistema de reconocimiento de voz sea complejo, en especial si se desea que este sea independiente del usuario. Dentro de estos problemas se encuentran:

- La continuidad en el habla.

El ser humano habla de forma continua y algunas veces es complicado para los sistemas determinar con exactitud donde comienza y donde termina una palabra dentro de una oración, esto se debe a que cada persona pronuncia con una cadencia diferente, es decir pronuncia las palabras con diferente velocidad, provocando que en ocasiones las palabras se mezclen o estén tan separadas que el sistema interprete ese espacio como el final de la oración.

Además de los problemas para delimitar las palabras los sistemas pueden confundirse con expresiones que se emiten cuando la persona está pensando que decir, cuando manifiesta su sorpresa o admiración, como por ejemplo "uh", "um."

- Cambios en la voz de una persona

La voz de una persona varía debido al tiempo y amplitud de los movimientos de las articulaciones involucradas en su producción. Es por ello que una misma palabra pronunciada por el mismo locutor en diferentes ocasiones o condiciones es esencialmente diferente, aún cuando para el ser humano estos no son perceptibles, los sistemas de reconocimiento sí son susceptibles a estos cambios.

Debemos tener en cuenta además de los cambios imperceptibles por las personas aquellos que si lo son, como los cambios debidos al estado de animo de una persona; es decir si esta fatigado, contento, enojado o los debidos a alguna enfermedad.

- Las variaciones en la voz de diferentes personas.

Si queremos que nuestro sistema sea capaz de reconocer palabras sin importar que usuario las pronuncie se debe tener en cuenta que las palabras presentan cambios significativos de una persona a otra. Dichos cambios se deben a diversos factores, como por ejemplo las variaciones de acento de una región geográfica a otra o las diferencias anatómicas que existen entre las cuerdas vocales del hombre y la mujer.

Las mujeres y los niños pronuncian las palabras en un tono más agudo, es decir su voz tienen componentes de frecuencia más alta que los hombres, esto se debe a que sus cuerdas vocales son más pequeñas. Las voces de mujeres y niños han sido menos estudiadas que las de los hombres debido a la gran dificultad para extraer sus parámetros, [3] esto representa una desventaja porque los métodos para obtener las características de la voz se basan en el aparato vocal masculino y por lo tanto se pierde precisión al analizar señales de voz con componentes de frecuencia alta.

- Número de elementos que componen las características de la voz y número de palabras a reconocer.

Los recursos de cualquier sistema son limitados, es por ello que se deben aprovechar al máximo. Uno de estos recursos es la memoria en la cual se deben almacenar las características de las palabras a ser reconocidas. Si se quiere tener una base grande de palabras existe la posibilidad de que nuestro sistema confunda unas con otras debido a la similitud entre ellas, además no importa que tan grande sea nuestra base, siempre habrá palabras que no hallamos contemplado.

Para evitar problemas de almacenamiento y confusión de una palabra con otra es necesario extraer de la señal de voz una serie de valores que nos proporcionen una buena representación de esta con pocos datos, esta representación debe ser lo suficientemente consistente para que no cambie cuando se pronuncien palabras iguales, eliminando de las señales de voz las características propias de cada individuo y que además nos permita distinguir palabras diferentes.

Si el número de características es menor, se reduce el espacio necesario de almacenamiento y se evita realizar cálculos sobre información redundante.

- Recepción de la señal e interferencia del ambiente.

El medio en el cual se graban y pronuncian las palabras a ser interpretadas por el SRV influye notablemente en la obtención de un sistema que sea capaz de reconocer con éxito las palabras, esto se debe a que la percepción de la voz cambia dependiendo de la acústica del cuarto empleado, del ruido ambiental, etc. El ruido ambiental se mezcla con la señal de la voz distorsionándola o en casos más extremos ocultándola por completo.

La calidad de la señal depende en gran medida de los instrumentos empleados para transformar la voz humana en señales eléctricas. La calidad de dichos instrumentos varía

desde micrófonos de alta calidad y gran ancho de banda hasta celulares en movimiento con recepción limitada, además diferentes micrófonos tienen tiempos de respuesta distintos por lo que la posición del mismo provoca también variaciones en la señal de voz.

Todos los problemas de recepción, interpretación e interferencia que se pueden presentar al enviar e intercambiar mensajes de un locutor a otro son compensados por el oído y el cerebro humano haciendo que esta interacción emisor-receptor sea de forma natural e instantánea. Aún con los avances tecnológicos actuales, los sistemas de reconocimiento de voz no han podido igualar las características y funcionalidad de ninguno de estos órganos provocando que la respuesta que dan sea de cierta forma limitada.

Como hemos visto son varios los factores que impiden obtener un sistema de reconocimiento de voz que responda adecuadamente a todas sus señales de entrada. Para obtener un sistema simple y barato que se desempeñe correctamente es necesario acotar nuestro problema, es decir limitar un poco al usuario en el empleo del mismo y de esta forma reducir su complejidad. Se puede por ejemplo pedir que los usuarios pronuncien las palabras de forma pausada (reconocimiento no continuo del habla), limitar el uso del sistema a un solo usuario (sistemas dependientes del usuario), limitar el número de palabras que el sistema puede reconocer, utilizar el sistema en lugares donde el ruido ambiental no sea excesivo, etc.

Si mantenemos constantes algunos parámetros minimizamos los efectos que estos puedan tener sobre el sistema y podemos concentrar nuestros esfuerzos en aquellos que, por especificaciones del problema no se puedan limitar. Por ejemplo si nuestro objetivo es desarrollar una aplicación de dictado el sistema solo requiere reconocer la voz de una sola persona y nuestro problema principal es aumentar el número de palabras que se desea reconocer.

En resumen, la problemática de los sistemas de reconocimiento de voz se puede dividir principalmente en cuatro partes:

1. Poder de cómputo.

   El tipo de sistema de cómputo empleado en el desarrollo y uso del sistema de reconocimiento de voz, es decir si se va a utilizar una computadora personal (PC), una estación de trabajo o una supercomputadora.

2. Número de usuarios.

   La cantidad de usuarios que nuestro sistema es capaz de reconocer. Existen dos tipos de sistemas, aquellos orientados a varios usuarios (sistema múlti-usuario) y los que son utilizados por un solo usuario.

3. Número de palabras a ser reconocidas y continuidad del habla

En este caso, se determina la cantidad de palabras que serán reconocidas por el sistema. Generalmente se dice que el sistema tiene un vocabulario reducido si este reconoce de 10 – 1000 palabras y un vocabulario grande si reconoce más de 1000. [3]

También hay que establecer si nuestro sistema reconocerá la voz en forma continúa o se tiene que pronunciar cada palabra separada por una pausa.

4. Calidad de los transductores

Es decir si el instrumento empleado para convertir la voz en una señal eléctrica será un micrófono conectado a la tarjeta de sonido de una computadora o el micrófono de un teléfono. También se debe tomar en cuenta si el lugar en el cual se empleará el sistema es muy ruidoso, ya que la presencia de un ruido excesivo influirá negativamente en el desempeño del sistema.

1.4 Reconocimiento de Voz con Redes Neuronales Artificiales.

Debido a las grandes ventajas de utilizar la voz como forma de interacción entre los seres humanos y las máquinas, muchos investigadores se han dedicado al estudio y realización de sistemas de reconocimiento de voz. Algunos de ellos han orientado sus estudios a la solución del reconocimiento de voz con redes neuronales artificiales. Dentro de éstos destacan Alex Waibel, Terrence Sejnowsky, Teuvo Kohonen y Richard Lippmann.

Existen muchos artículos importantes relativos al reconocimiento de voz con redes neuronales artificiales, en ellos se aplican topologías y métodos de entrenamiento muy variados. Desde perceptrones multicapa entrenados con el algoritmo de retropropagación del error hasta redes que combinan la topología del perceptrón con los modelos de Marcov.

Son muchos los problemas relacionados con el reconocimiento de voz en los cuales se han utilizado redes neuronales. Dentro de ellos se encuentran los trabajos realizados por Waibel para crear sistemas que traduzcan palabras de un idioma a otro de forma automática [25], otros como los efectuados por Lippman hacen énfasis en el procesamiento necesario para obtener las características de la voz que serán utilizadas como entradas de la red neuronal [13].

A continuación se presentan de forma breve, dos ejemplos en los cuales fueron utilizadas las redes neuronales artificiales para la clasificación de palabras:

- Reconocimiento continuo de palabras utilizando redes LPNN.

Alex Waibel presentó en su artículo "Large Vocabulary Recognition Using Linked Predictive Neural Network" [24] un sistema que empleaba un tipo de red llamada Linked Predictive Neural Network (LPNN) para reconocer palabras en japonés.

Waibel señala que una red LPNN no realiza reconocimiento por clasificación, sino por predicción. Y la idea básica en este tipo de redes es la siguiente: La red toma como entrada determinados segmentos de la señal de voz, los pasa a su capa oculta y trata de predecir el siguiente segmento. La predicción realizada por la red es comparada con el segmento actual, si el error es pequeño se considera que la red es un buen modelo para ese segmento de la señal.

Siguiendo el esquema anterior, Waibel determinó que si se pudiera enseñar a la red a hacer predicciones precisas durante los segmentos correspondientes a un fonema (unidad mínima en la que pueden dividirse las palabras de un idioma) y que no respondiera a cualquier otro segmento de la señal de voz, entonces se obtendría un red capaz de reconocer de forma efectiva ese fonema en particular. Para lograr este fin asignó a cada fonema tres redes neuronales, una para analizar el principio, otra para la mitad y la tercera para el final.

Una palabra es representada entonces por una secuencia lógica de los fonemas que la forman. Por ejemplo la palabra "oso" es representada por la secuencia de fonemas: o, s, o. A cada uno de estos fonemas se le asignan tres redes que analizan su principio, duración y final. Las ocurrencias múltiples de la redes como en caso de la letra "o" se toman como una sola debido a que hay un solo modelo para cada fonema.

Con esta topología de red se realizaron pruebas cuyo fin era reconocer un conjunto de palabras en japonés. Las características de las palabras que fueron empleadas como entrada de la red se obtuvieron con el método de los coeficientes ceptrum en escala mel. A diferencia del perceptrón las entradas de la redes LPNN son dinámicas, por lo que se calculaban los coeficientes ceptrum cada 10 ms.

Las palabras capturadas se dividieron en dos conjuntos, uno formado por palabras fonéticamente iguales y otro por palabras diferentes. El primer conjunto estuvo compuesto por 229 palabras de ocho fonemas y el segundo se formó con 900 palabras de catorce fonemas.

Como parámetro de comparación se utilizó la eficiencia, la cual se define como el número de patrones de prueba reconocidos satisfactoriamente por la red entre el número total de patrones de prueba involucrados. Los valores de eficiencia que Waibel obtuvo fueron los siguientes:

Palabras similares (229 palabras)	Palabras diferentes (900 palabras)
0.94	0.9

Tabla 1.1. Eficiencias obtenidas en las pruebas realizadas por Waibel para palabras en idioma japonés

Como se puede observar en la tabla anterior los valores de eficiencia que se obtuvieron fueron grandes, aún cuando el numero de palabras se aumentó a más del triple.

- Reconocimiento palabras aisladas utilizando un perceptrón multicapa.

Ricardo Zebulum, Marley Vellasco y Guy Perelmuter en su articulo "A Comparison of Different Spectral Analysis Models for Speech Recognition Using Neural Networks" [26], analizan el uso de las redes neuronales artificiales entrenadas con el algoritmo de retropropagación del error para reconocer diez palabras aisladas pronunciadas en portugués.

En su trabajo Zebulum y sus colegas realizaron pruebas con tres tipos de coeficientes que representan las señales de voz, estos son: el método de los coeficientes de predicción lineal, el método de los coeficientes ceptrum en escala mel y como tercero utilizaron una combinación de los dos anteriores. Los tres métodos mencionados no fueron implementados por los autores del artículo, por lo que para obtener dichos coeficientes tuvieron que utilizar un programa desarrollado por la universidad de Oregon llamado OGI speech tools [22].

Las señales de voz que utilizaron tanto en la fase de entrenamiento como en la fase de prueba de la red, fueron capturadas en un ambiente ruidoso con el fin de simular las condiciones reales en las cuales trabajaría su sistema.

El número de personas con los que se formaron los conjuntos de prueba y entrenamiento fue de catorce, de los cuales nueve se usaron como conjunto de entrenamiento y los cinco restantes como patrones de prueba.

Por cada uno de los tres tipos de coeficientes se utilizaron dos conjuntos de entrenamiento: El primer se formó con tres repeticiones de cada palabra, mientras que en el segundo se utilizaron solo dos repeticiones. Como se tienen diez palabras diferentes, el primer conjunto se contenía un total de 270 palabras (10 palabras x 9 personas x 3 veces) y el segundo 180.

La red neuronal utilizada fue un perceptrón con una capa oculta y el método de entrenamiento fue el algoritmo de retropropagación del error. La capa de salida estuvo compuesta por diez neuronas, cada una representando una palabra.

En la siguiente tabla se muestran los valores de eficiencia que obtuvieron los autores del artículo:

Tipo de Coeficiente	Caso 1 (tres repeticiones)	Caso 2 (dos repeticiones)
MFCC	0.72	0.82
LPC	0.72	0.80
Híbrido	0.74	0.84

Tabla 1.2. Valores de eficiencia que obtuvieron en promedio para cada tipo de coeficientes. MFCC significa coeficientes ceptrum en escala mel, LPC coeficientes de predicción lineal e Híbrido hace refiere a los dos anteriores.

Todo el proceso de reconocimiento, desde la captura de las palabras hasta la clasificación de las mismas se realizó en una estación de trabajo SUN SPARC. Con este sistema de cómputo el entrenamiento de las redes fue de 24 horas.

Además de las pruebas anteriores los autores del artículo realizaron una prueba adicional, dicha prueba consistió en agregar otra capa oculta a la red para ver si la eficiencia mejoraba, sin embargo esto no se logró y el tiempo requerido para el entrenamiento fue mayor.

Debido a que en este ejemplo se utiliza la misma arquitectura de red neuronal y la misma representación de las señales de voz empleadas en esta tesis, este artículo nos será de utilidad para tener un punto de comparación con los resultados obtenidos en el presente trabajo.

Los dos artículos anteriores son un ejemplo de las investigaciones realizadas en el campo del reconocimiento de voz, como se puede apreciar la arquitectura de las redes empleadas en este tipo de problemas pude ser variada.

Cabe destacar que a medida que queramos que el sistema de reconocimiento abarque un número mayor de palabras el equipo de computo necesario para realizar el entrenamiento debe ser más poderoso debido a la cantidad de operaciones que deben realizarse y al almacenamiento de datos.

CAPITULO II

REDES NEURONALES ARTIFICIALES

REDES NEURONALES ARTIFICIALES.

El cerebro es un órgano que coordina nuestros movimientos y nos permite realizar nuestras actividades con base al conocimiento acumulado desde la niñez, además de permitirnos aprender de nuevas experiencias y almacenar recuerdos.

El cerebro esta compuesto por billones de neuronas que se encuentran interconectadas entre si formando una compleja estructura que desde hace siglos el hombre a tratado de entender. Para comprender el funcionamiento del cerebro es necesario conocer bien la intrincada red de conexiones de células nerviosas que lo forman y como estos elementos interactúan entre si para lograr que el conocimiento se almacene en el cerebro o esté realice operaciones.

El estudio del cerebro es extremadamente complejo debido a la gran cantidad de neuronas que lo forman. Por muchos años los primeros anátomistas tuvieron que desgarrar el tejido nervioso para obtener neuronas aisladas, este trabajo se redujo gracias al método de coloración de Golgi [11], el cual permite obtener perfectas y definidas imágenes de la neurona y sus prolongaciones. El método de Golgi consiste en fijar el tejido nervioso con sustancias tales como el ácido ósmico y bicromato de potasio, para posteriormente aplicar como colorante nitrato de plata. Con este método se tiñen solamente unas pocas neuronas de todo el tejido permitiendo obtener células aisladas enteras.

El método de coloración de Golgi, junto con la invención de microscopios más potentes, como el microscopio electrónico han facilitado el estudio del cerebro, motivando a más investigadores a realizar trabajos orientados a descifrar la forma en la cual las neuronas realizan sus operaciones.

Dentro de los estudios más sobresalientes en el campo de la fisiología celular se encuentran los desarrollados por el doctor Santiago Ramón Cajal en 1911, quien propuso que las neuronas eran los elementos fundamentales que componen el cerebro, Cajal además estableció modelos de neuronas de diferentes partes del cuerpo los cuales han sido tomados como base para comprender su funcionamiento.

La velocidad en la cual las neuronas realizan sus tareas es lenta comparada con la velocidad con la que las computadoras llevan a cabo sus operaciones, sin embargo estas máquinas no han logrado reconocer patrones visuales o sonoros de forma tan precisa y en los tiempos tan pequeños en los que el cerebro lo hace. Esto se debe fundamentalmente a que la cantidad de información contenida en todos los elementos de almacenamiento del cerebro (las neuronas) supera por mucho la capacidad actual de almacenamiento en disco de las computadoras, además de que el cerebro distribuye su trabajo entre todos sus elementos, es decir, cada neurona que contiene el cerebro se encuentra realizando sus propias operaciones al mismo tiempo que las demás lo hacen, esta forma de división del trabajo se conoce como cálculo en paralelo.

Los procesos en paralelo permiten realizar muchas operaciones simultáneamente, a diferencia de los procesos secuenciales con los que se puede hacer un solo cálculo a la vez. Si tomamos en cuenta que existen millones de neuronas el número de operaciones que el cerebro hace por segundo es muy grande.

Además de las limitaciones computacionales cabe recordar que no se conoce con exactitud el funcionamiento del cerebro, por lo tanto los modelos que se tienen de él no son todavía representativos sino más bien aproximados y burdos.

Los primeros trabajos desarrollados para encontrar un modelo del funcionamiento del cerebro fueron planteados durante la década de los 40's y principios de los 50's por varios matemáticos, biólogos y psicólogos entre los que destacan McCulloch y Pitts [5], Housseholder y Landahl y Kleene quienes desarrollaron modelos matemáticos de neuronas. Posteriormente, investigadores como Farley, Clark, Rochester, Holland y Habit desarrollaron las primeras simulaciones de redes neuronales en computadora.

En 1962 Rosenblatt publicó su libro de *"Principles of Neurodynamics"* en el que presentó formalmente al Perceptrón como modelo para construir redes neuronales. [12] Rosenblatt desechó la ideal que se tenía hasta ese entonces de que las Redes neuronales eran un modelo que representaba correctamente el funcionamiento del cerebro, en su lugar propuso que estas fueran empleadas como asociadores y clasificadores de patrones. Con la aparición del Perceptrón se solucionaron dos problemas: definir una función que midiera el error y encontrar un procedimiento que redujera ese error por medio de un ajuste en sus parámetros (regla de aprendizaje). La red propuesta por Rosenblatt es considerada la primera red neuronal artificial capaz de aprender.

Los modelos computacionales conocidos como Redes Neuronales Artificiales (RNA) son modelos estadísticos que están inspirados en las características fisiológicas del cerebro. A grandes rasgos una RNA es una estructura computacional compuesta por elementos procesadores interconectados entre si realizando cálculos en forma paralela. Estos elementos procesadores son usados para estimar las características de un conjunto de datos utilizando un número pequeño de ejemplos de los cuales pueda "aprender", tratando de imitar con ello la forma de operación del cerebro.

Al descubrirse que las RNA no eran capaces de resolver problemas tan simples como la simulación de una compuerta OR exclusivo, muchos investigadores de la década de los 60's que se encontraban trabajando con redes neuronales perdieron su interés en ellas, provocando que su desarrollo se "estancara" por varios años, sin embargo Teuvo Kohonen, Hopfield, Stephen Grossberg y James Anderson encontraron modelos que superaron las limitaciones encontradas, dando con ello un nuevo impulso al desarrollo de las RNA.

La ingeniería enfrenta cotidianamente problemas en los cuales los datos están incompletos o se encuentran mezclados con otras señales como por ejemplo el ruido, en este tipo de situaciones es importante realizar predicciones razonables acerca de los datos perdidos, esta es una tarea difícil y es aquí donde las redes neuronales son útiles.

Actualmente, las redes neuronales han probado ser eficientes en problemas complejos, difíciles de resolver con algoritmos secuenciales tales como el reconocimiento de patrones, reconocimiento de imágenes y del habla, el control de procesos y el procesamiento de señales. [4] Dentro de las ventajas que han motivado el empleo de redes neuronales en este tipo de problemas podemos mencionar las dos siguientes:

1- Dado que las RNA son una estructura compuesta por elementos que realizan cálculos masivos en paralelo son candidatas a emplearse en algoritmos que requieran una respuesta rápida.
2- No es necesario reprogramar el sistema para que esté sea capaz de reconocer nuevos patrones. Basta con entrenar a la red con los nuevos patrones , actualizar los pesos e integrarlos de nuevo en la aplicación.
3- Debido a que las redes neuronales artificiales utilizan generalmente funciones no lineales para generar sus salidas, estas son útiles para resolver una gran variedad de problemas, principalmente aquellos en los cuales las entradas son no lineales.

2.1 Neurona Biológica.

Para comprender la arquitectura de una RNA es preciso tener nociones de la estructura, funcionamiento e interrelación que existe entre los elementos biológicos en el que se basan: las neuronas.

Una neurona es una célula perteneciente al sistema nervioso central que tiene como función almacenar información y transmitirla a otras neuronas. Las partes que la constituyen se muestran en el diagrama de la figura 2.1.

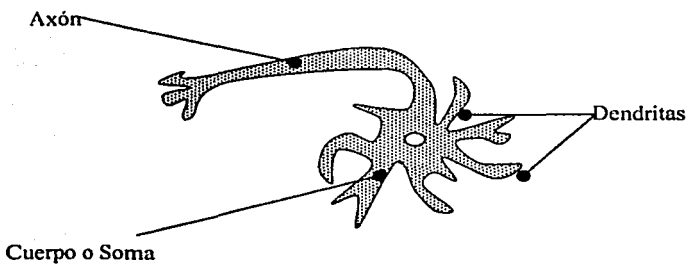


Figura 2.1. Partes constituyentes de una neurona biológica.

Como se aprecia en el diagrama anterior, una neurona se compone de un cuerpo celular o "soma" del cual surgen pequeñas ramificaciones llamadas dendritas que funcionan como receptores de señales provenientes de otras neuronas o terminales nerviosas ubicadas en todo nuestro cuerpo. Además de las dendritas la neurona posee una

ramificación mayor llamada axón la cual funge como terminal de salida, para enviar su salida a otras neuronas el axón se ramifica.

Cuando el axón de una neurona emisora se conecta con las dendritas de una neurona receptora se forma una unión por la cual se transmiten señales de una neurona a otra, esta unión recibe el nombre de sinapsis. La comunicación entre neuronas se realiza por medio de impulsos eléctricos que duran alrededor de 1 ms, este impulso también denominado potencial de acción se mueve con rapidez a lo largo del axón llegando a la neurona receptora a través de las sinapsis. Cada impulso recibido por la sinapsis inicia la liberación de una pequeña cantidad de sustancia química denominada sustancia transmisora que es la que viaja a la neurona siguiente, donde esta el receptor.

La señal emitida por una neurona es producto de un proceso que esta realiza sobre los impulsos eléctricos que recibe. El proceso de emisión es controlado por un potencial interno asociado a la neurona. Si este potencial supera un cierto umbral denominado umbral de disparo, se envía un impulso eléctrico al axón. En caso contrario, no se envía ninguna señal y se dice que la neurona esta en reposo.

Una neurona está polarizada debido a un potencial interno negativo en el interior de la membrana celular respecto al exterior. Esto se origina por la libre circulación de iones de potasio con carga positiva a través de la membrana celular y al mismo tiempo a la retención de moléculas con carga negativa dentro de la célula.

Las señales que recibe una neurona en la sinapsis son de dos tipos: excitadoras e inhibitorias. Una señal excitadora contribuye positivamente al potencial interno de la neurona, volviéndolo menos negativo, provocando con ello que dicho potencial se acerque más al umbral de disparo. Por otra parte, una señal inhibitoria mantiene el potencial interno de la neurona por debajo del umbral de disparo.

Dado que la neurona recibe varias señales excitadoras e inhibitorias. La neurona tiene que determinar el valor neto de entrada para establecer si el potencial interno sobrepasa el umbral de disparo. Esto se realiza por medio de la suma de todas las señales de entrada, proceso conocido como agregación.

2.2 Neurona artificial.

Aún cuando la unidad básica de una Red Neuronal Artificial suele llamarse neurona en alusión a los elementos del sistema nervioso central, cabe recordar que estás no son una representación exacta de las neuronas del cerebro.

Una neurona artificial o simplemente neurona es una unidad procesadora que tiene los siguientes cuatro elementos funcionales:

- uno o más receptores
- un sumador
- una función de activación
- una sola salida.

La forma en la que se relacionan estos elementos es la siguiente:

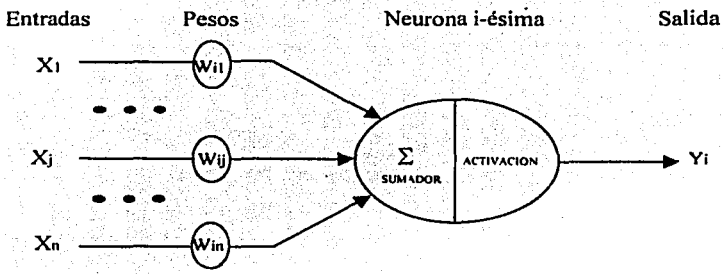


Figura 2.2. Representación de una neurona artificial

Los receptores como su nombre lo indica tienen la finalidad de recibir las señales de entrada X_i que provienen por lo general de las salidas de otras neuronas.

Cada conexión existente entre las neuronas tiene asociada una magnitud llamada peso o intensidad de conexión. Siguiendo la representación de la figura 2.2 el valor del peso es un número real que se denota por medio de las letras W_{ij}, en donde los subíndices ij indican las neuronas a las que esta asociado dicho peso, siendo i la neurona que recibe la señal y j la neurona que envía la señal. El peso de conexión determina el estado en el cual se encuentra la neurona, si hacemos una analogía con la neurona biológica un valor positivo de W_{ij} puede ser visto como una excitación y uno negativo como una inhibición de las conexiones sinápticas de las neuronas del cerebro.

Los pesos de las conexiones pueden ser agrupados en forma de un arreglo matricial, esto es importante pues permite que todos los pesos sean manejados fácilmente por medio de algoritmos computacionales.

Dado que la neurona recibe varias entradas, esta tiene que obtener un valor neto de entrada y con base a él generar una sola salida que sea transmitida a todas las neuronas a los que está conectada. El valor neto de entrada, también conocido como *actividad lineal de la neurona* se obtiene por medio de una suma ponderada del producto de las señales de entrada por los pesos de sus conexiones. Lo anterior se representa matemáticamente por medio de la expresión:

$$\textit{actividad lineal de la neurona i-ésima} = \sum_{j=1}^{n} X_j W_{ij} \qquad 2.1$$

Donde X_j es el j-ésimo patrón de entrada, W_{ij} es el peso de la conexión que existe entre la entrada X_j y la neurona i-ésima y n es el número de conexiones que tiene la i-ésima neurona.

La salida de la neurona se obtiene por medio de una función de activación, la cual toma como parámetro la actividad lineal de la neurona calculada con la expresión 2.1, es decir:

$$Y_i = f\left(\sum_{j=1}^{n} X_j W_{ij}\right) \qquad 2.2$$

en donde $f(x)$ es la función de activación.

La salida de la neurona indica el estado en el que esta se encuentra, es decir está activa (excitada) o no (inhibida).

2.3 Función de activación

La función de activación es una función generalmente no lineal cuyo propósito es determinar con base al valor de la actividad lineal de la neurona, el estado en el que esta se encuentra, es decir si la neurona se activa excitando a otras neuronas o no. Además de determinar el estado de la neurona, la función de activación mapea o limitan los valores que puede generar la neurona a su salida.

Dentro de las funciones de activación empleadas en el modelo de las RNA se encuentran las siguientes:

- Funciones lineales.

La función lineal más utilizada es la función identidad. Con este tipo de función la actividad lineal de la neurona se considera como su propia salida.

- Funciones escalón.

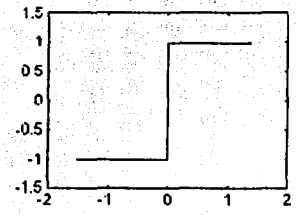
Son funciones que dan una salida binaria la cual depende de si el valor de entrada está por encima o por debajo del valor de umbral. Ejemplos de estas funciones son la función signo y la escalón estándar definidas como:

Función Signo

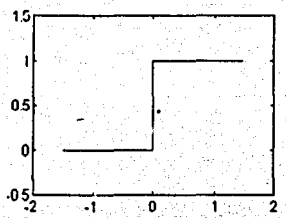
$$sgn(x) = \begin{cases} -1 \ si \ x < 0 \\ 1 \ si \ x \geq 0 \end{cases}$$

Función Escalón

$$F(x) = \begin{cases} 0 \ si \ x < 0 \\ 1 \ si \ x \geq 0 \end{cases}$$



(a)



(b)

Figura 2.3. En la figura (a) se muestra la gráfica de la función signo y en la (b) se muestra la gráfica de la función escalón.

Este tipo de funciones determinan un umbral objetivo de disparo para la actividad de la neurona.

- Funciones sigmoidales.

Son funciones monótonas acotadas que dan una salida gradual no lineal para las entradas. Dentro de ellas se encuentran la logística y la tangente hiperbólica definidas como:

Sigmoide logística

$$f(x) = \frac{1}{1 + e^{-cx}}$$

Sigmoide tangente hiperbólica

$$f(x) = \tanh(cx)$$

En ambos casos la constante c es mayor que cero 0 y tiene influencia directa en la pendiente de la función sigmoide.

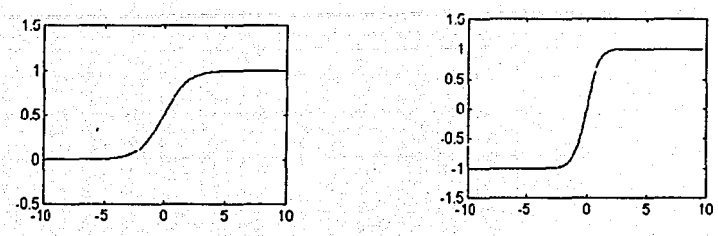


Figura 2.4. En la figura (a) se encuentra la gráfica de la función Sigmoide logística de 0 a 1, mientras que en la (b) se muestra la función sigmoide tangente hiperbólica entre –1 y 1.

Además de las funciones anteriores existen otras que se utilizan como funciones de activación de las neuronas de una RNA, como por ejemplo la función linear a tramos, la gaussiana o la sinusoidal, sin embargo, las más utilizadas en la fase de entrenamiento, principalmente para el algoritmo de retropropagación del error (el cual se empleará en esta tesis) son las funciones sigmoide debido a que son continuas, invertibles y diferenciables en todo su dominio [10], a diferencia de las funciones escalón y signo.

Como la función de activación tiene la labor de convertir el valor de cada neurona en su correspondiente salida, se emplea un valor de umbral para determinar si la neurona dispara o no. Es decir, la neurona estará activa si su actividad lineal sobrepasa cierto valor de umbral:

$$\text{Salida de la red} = f\left(\sum_{j=1}^{n} X_j W_{ij}\right) \quad \text{si} \quad \sum_{j=1}^{n} X_j W_{ij} > \text{valor de umbral}$$

en caso de no superarse el umbral la salida de la red sería cero.

2.4 Topologías de Redes Neuronales Artificiales.

Una RNA es un conjunto de neuronas artificiales del tipo mostrado en la figura 2.2 que se conectan entre si formando una estructura de red. La forma en la que estas neuronas se conectan depende de la arquitectura. Dentro de las arquitecturas más empleadas se encuentran:

- Perceptrones Multicapa (PMC) o redes de conexión hacia adelante. En esta topología las neuronas de una capa se conectan con las neurona de la siguiente, pero no existen conexiones entre neuronas de la misma capa.

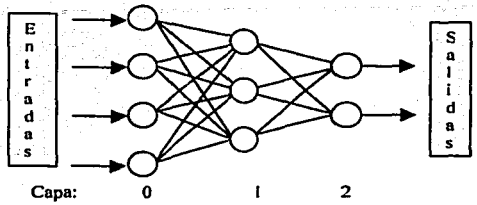


Figura 2.5. Perceptrón multicapa (PMC) con 4 neuronas en la capa de entrada, 3 en la capa oculta y dos en la salida. Las líneas indican la forma en la cual se conectan las neuronas.

- Redes Recurrentes de Hopfield o de conexiones laterales. Las entradas de este tipo de redes además de provenir del exterior, provienen también de sus mismas salidas. En esta topología las neuronas de la misma capa se conectan. En la siguiente figura se muestra una red neuronal de Hopfield con tres neuronas.

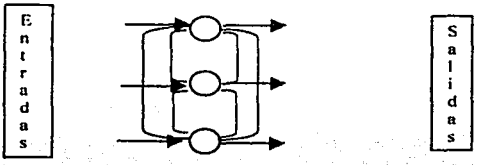


Figura 2.6. Red recurrente de Hopfield con tres neuronas. En este caso las neuronas de la misma capa se conectan entre si.

- Redes Auto organizadas de Kohonen o de conexiones hacia atrás. En este caso las neuronas de una misma capa compiten entre si para tener la mayor actividad.

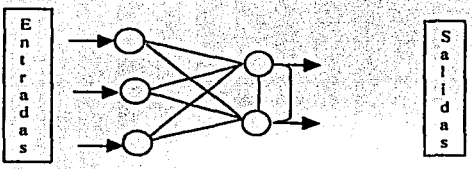


Figura 2.7. Red Auto organizada de Kohonen de dos capas y capa de salida competitiva.

Algunas topologías de redes neuronales artificiales son más utilizada que otras para la solución de un problema específico, una de las topologías utilizadas en problemas de reconocimiento de voz es la del Perceptrón multicapa [6].

El perceptrón multicapa es una extensión de la red propuesta por Rosenblatt. Esta compuesta por neuronas agrupadas en capas numeradas de 0 a N, en donde el número de capa indica la distancia a la que se encuentran sus neuronas respecto a las neuronas de la capa de entrada. La capa número 0 es la capa de entrada, la capa N es la de salida y las restantes son capas intermedias denominadas capas ocultas. En la figura 2.5 se muestra una red con tres capas.

Las neuronas de la capa de entrada simplemente transmiten sus valores a todas y cada una de las neuronas de la capa oculta sin realizar ningún cálculo. Por otro lado las neuronas de la capa de entrada y de la capa oculta interactúan para conseguir los valores de entrada de la capa de salida. Después de que las neuronas de la capa de salida reciben sus entradas, estas generan la respuesta final de la red. El número de neuronas de la capa de salida lo determina la respuesta real que deseamos obtener de la red a los datos de entrada, mientras que las neuronas de la capa de entrada y las neuronas de la capa de salida están especificados por el problema a resolver. Para establecer el número de elementos de la capa oculta no existe una regla bien definida, por lo que se debe elegir un valor inicial, el cual es modificado a fin de obtener el mejor desempeño de la red.

2.5 Entrenamiento.

Después de elegir la topología que se adecué a nuestro problema, el siguiente paso consiste en entrenar la red para que esta adquiera "conocimiento" y lo almacene en sus pesos. El entrenamiento de una red consiste en ajustar los pesos de conexión de sus neuronas de tal forma que en ellos se encuentre codificada la información contenida en los datos de entrenamiento. Una RNA utiliza un proceso de "aprendizaje" por analogía, es decir se debe alimentar a la red con datos representativos que puedan ser tomados como ejemplo del problema a resolver.

El proceso de entrenamiento de una RNA esta inspirado en la forma en la cual las neuronas biológicas ajustan sus potenciales internos, cambiando el estado de sus sinapsis y la influencia de una neurona sobre otra.

El ser humano ajusta los pesos de las conexiones sinápticas de su cerebro "ejercitándolo" a través de ejemplos que le permitan adquirir conocimiento. Supongamos el caso de un niño que sabe pronunciar las vocales pero no sabe como escribirlas, para que pueda aprender el significado de las letras es necesario mostrarle dichas letras en forma escrita y pronunciarlas para que el niño pueda asociar el sonido de la vocal con su símbolo gráfico, este proceso se repite hasta que el niño es capaz de asociar y clasificar correctamente cada tipo de letra. Cada vez que se le presenten nuevas letras, escritas con diferente tamaño, color o forma, el niño será capaz de reconocerlas. Si bien esta forma de

aprendizaje es natural y ha sido empleada por el hombre desde hace miles de años, este no ha sido capaz de crear una máquina que reconozca todos los patrones que se le presenten y que además los asocie de forma eficaz y sencilla.

Uno de los primeros métodos de entrenamiento fue formulado por Donald Hebb en su libro "Organization of Behavior" [9] en esté libro se propone que los pesos de las conexiones se deben ajustar con base a los valores de los nodos que conecta.

Los métodos de aprendizaje se pueden clasificar en dos categorías dependiendo del tipo de información que se les proporciona, estos son entrenamiento, supervisado y no supervisado.

- Entrenamiento no supervisado.

En este tipo de entrenamiento los datos se presentan a la red de forma desordenada para que sea la red la que clasifique estos patrones en categorías, está propiedad es conocida como auto-organización. Algunos de los algoritmos de aprendizaje no supervisado son:

- Aprendizaje Hebbiano. Esta inspirado en el modelo de refuerzo de las sinapsis neuronal propuesto por Hebb. En primer lugar se eligen pesos aleatorios iniciales, estos pesos se modifican con base a la correlación existente entre los valores de entrada y el error resultante, esto se expresa por medio de la siguiente ecuación:

$$\Delta W_{ij} = - \eta \, (Salida_{deseada} - Salida_{real}) \, . \, Entrada \qquad 2.3$$

donde el parámetro η es un factor de aprendizaje (learning rate) el cual nos indica la razón de cambio de los pesos, es decir controla la velocidad de aprendizaje de la red.

- Entrenamiento supervisado.

En los entrenamientos supervisados existen dos conjuntos de datos, uno formado por los patrones de entrada y el otro compuesto por valores de salida deseados. En este tipo de entrenamiento los pesos se obtienen minimizando el error resultante de una comparación entre la salida proporcionada por la red y la salida deseada. Este error se obtiene por medio de alguna función de error como por ejemplo la función de los minimos cuadrados. Un ejemplo de entrenamiento supervisado es la regla delta.

- Método de descenso de gradiente o regla delta. Al igual que en el aprendizaje Hebbiano, se eligen pesos aleatorios. La meta de este método es minimizar la suma de los cuadrados de los errores:

$$E = \frac{1}{2}\sum (Salida_{deseada} - Salida_{real})^2 \qquad 2.4$$

El algoritmo consiste en "moverse" en sentido de disminución del gradiente, modificando iterativamente cada uno de los pesos W_{ij} por medio de un incremento ΔW_{ij} proporcional al gradiente de error, es decir:

$$\Delta W_{ij} = -\frac{\partial E}{\partial W_{ij}} = -\eta \sum (Salida_{deseada} - Salida_{real}) f'(a.l.) \cdot Entrada \qquad 2.5$$

donde *a. l.* es la actividad lineal de la neurona *i*-ésima obtenida con la ecuación 2.1, el parámetro η es el factor de aprendizaje y $f'(a.l.)$ es la derivada de la función de activación.

La derivada de la función de activación se emplea con el fin de "escalar" el error y forzar una corrección más grande cuando la suma ponderada queda cerca de la zona de pendiente máxima de la curva sigmoidal.

2.6 Retropropagación.

Otro algoritmo empleado en entrenamientos supervisados es el conocido como Retropropagación del Error (RP) o Backpropagation, este método permite ajustar los pesos de las conexiones de la red con base a la diferencia que existe entre la salida real y la deseada. Además permite ajustar los pesos de las conexiones que existen entre la capa oculta y la capa de salida. La popularidad del método radica en que es fácil de entender e implementar, además permite ajustar de una forma relativamente fácil los pesos de las neuronas de las capas ocultas.

La primera propuesta del método de retropropagación fue hecha por Paul Werbos en 1974 en su tesis doctoral, sin embargo el hecho no tuvo demasiada repercusión en su época y fue hasta el año de 1985 en el que esta idea fue retomada por David Rumelhart, Geofey Hinton y Ronald Williams en su libro *"Parallel Distributed Processing: Explorations in the Microstructures of Cognition"*, con el cual popularizaron el método de retropropagación del error presentándolo a la comunidad científica como una técnica útil de solución de problemas no lineales. [12]

El algoritmo de retropropagación involucra dos fases: La retropropagación hacia adelante y la retropropagación hacia atrás (de la cual toma su nombre).

Con base en la figura 2.8 establecemos el algoritmo de retropagación de la siguiente forma:

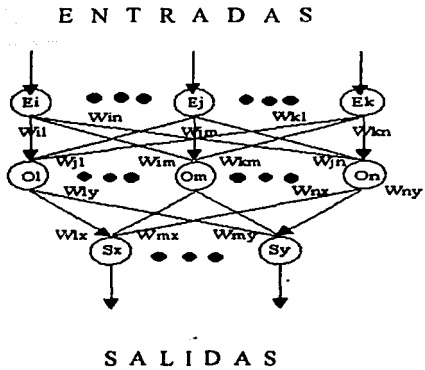


Figura 2.8. Red neuronal con "k" neuronas en la capa de entrada, "n" en la capa oculta y "y" neuronas en la capa de salida.

donde:

Ek	Es la neurona k-ésima de la capa de entrada	(capa 0).
On	Es la neurona n-ésima de la capa oculta	(capa 1).
Sy	Es la neurona y-ésima de la capa de salida	(capa 2).
W	Es el peso de la conexión existente entre la neurona de una capa y la neurona de la capa superior.	

En la fase de retropropagación hacia adelante se alimenta cada neurona de la capa de entrada E con su valor asociado del vector de entrada, estas neuronas no realizan cálculos, simplemente pasan sus valores a cada una de las neuronas de la capa intermedia.

Cada neurona de la capa intermedia genera una salida con base a la suma ponderada de sus entradas y a la función no lineal de activación. Empleando la ecuación 2.2:

$$O_n = f\left(\sum_{j=1}^{k} E_j W_{nj}\right)$$

esta salida es a su vez entrada de las neuronas de la capa de salida las cuales emplean también la ecuación 2.2 para producir la salida final de la red:

$$S_x = f\left(\sum_{j=1}^{n} O_j W_{xj}\right)$$

La fase de propagación hacia delante termina una vez que se han calculado toda las salidas S.

La fase de retropropagación hacia atrás es una generalización de la regla delta de Hebb, es decir se aplica a cada una de las conexiones la ecuación (2.5). Esta fase comienza con la comparación de la salida real de la red con la salida deseada, como resultado de esta comparación se obtiene un error, el cual es propagado en sentido inverso al de la propagación hacia adelante, es decir de la capa de salida a la de entrada a través de los pesos de conexión de las neuronas de la capa oculta.

Las neuronas de la capa intermedia estiman su error por medio de una suma ponderada del error que reciben de las conexiones con las neuronas de capa de salida, obtenidos en la fase de retropropagación hacia adelante. Si una unidad de salida tiene un error grande, entonces el error para las neuronas de la capa oculta también será grande.

Todos los pesos de la red se actualizan en proporción al error antes de comenzar nuevamente la fase de propagación hacia adelante, este esquema suele denominarse aprendizaje por lotes (batch). [14]

El procedimiento anterior se realiza hasta que el error se encuentre dentro de un rango previamente establecido.

En resumen el algoritmo de retropropagación es el siguiente:

1- Se establecen aleatoriamente todos los pesos iniciales.

2- Para cada patrón de entrada E_l calcular las salidas de las neuronas de la capa oculta $O_n = f\left(\sum_{j=1}^{k} \mathrm{Ej}\, \mathrm{W_{nj}}\right)$. que en forma matricial se expresa como: $[o_n] = f([E][W_n]')$, donde $[o_n]$ representa el vector de salida de la capa oculta, $[E]$ es el vector de entra y $[W_n]'$ es la matriz transpuesta de los pesos entre las neuronas de las capas de entrada y la capa oculta.

3- Para cada neurona de la capa oculta O_n calcular sus salidas: $[S_r] = f([O][W_x]')$ donde $[S_r]$ es el vector de salida de la última capa, $[O]$ es el vector obtenido en el paso 2 y $[W_x]'$ es la matriz transpuesta de los pesos entre las neuronas de la capa oculta y la capa de salida.

4- Comparar la salida de la red con la salida deseada. $(Salida_{deseada} - Salida_{real})$.

5- Actualizar los pesos de las conexiones de la capa de salida hacia la capa de entrada por medio de la ecuación 2.5.

6- Repetir los pasos 2 al 5 hasta que el error sea menor a un valor previamente especificado.

Gráficamente el algoritmo de retropropagación es:

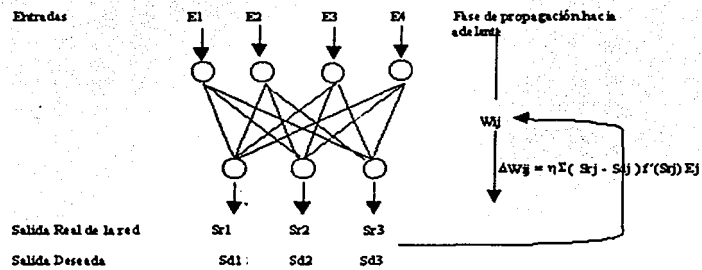


Figura 2.9. Las dos fases del algoritmo de retropropagación del error.

El algoritmo de retropropagación es lento, pero esto se puede mejorar agregando a cada capa una entrada de tendencia. Dicha entrada de tendencia es conocida también como neurona de bias y es tratada como una neurona mas, pero a diferencia de las demás neuronas de la capa esta no recibe ninguna señal de entrada, sino que mantiene un valor constante positivo y generalmente igual a +1.0 .

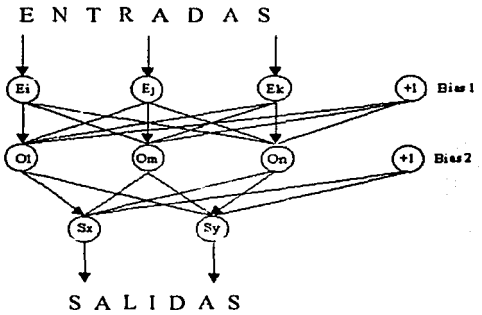


Figura 2.10. Perceptrón multicapa con dos neuronas de bias

Agregando una neurona de bias tanto en la capa de entrada como en la de salida, como se muestra en la figura 2.10, la salidas de las neuronas de la capa oculta y de salida se transforman en:

$$O_n = f\left(\sum_{j=1}^{k} E_j W_{nj} + Bias_1\right)$$

$$S_x = f\left(\sum_{j=1}^{n} O_j W_{xj} + Bias_2\right)$$

y todas las demás ecuaciones se deben modifican para considerar el término de bias.

Es importante comenzar con pesos iniciales aleatorios, porque si se parte de pesos nulos, el aprendizaje no progresa, esto se debe a que las salidas de las neuronas serán siempre nulas y el incremento en los pesos también, provocando que todos los pesos sean actualizados por la misma cantidad. [14] Se recomienda que los pesos iniciales además de ser aleatorios sean pequeños debido a que la actualización de los pesos es proporcional a la derivada de la función de activación; si se eligen valores grandes la derivada tiende más lento a cero, teniendo como consecuencia un entrenamiento lento.

Un aspecto muy importante en el algoritmo de retropropagación es el valor del factor de aprendizaje, si su valor es grande los cambios en el entrenamiento son más rápidos, provocando que la red no sea capaz de ajustar los pesos adecuadamente. El valor del factor de aprendizaje también influye en si la red alcanza una solución estable o no, si su valor es muy grande entonces las actualizaciones de los pesos no se aproximan al error mínimo provocando una oscilación en los pesos.

Existen en el mercado programas especializados en el entrenamiento de redes neuronales que incluyen el algoritmo de retropropagación del error como es el caso de DYNAMIND o que incluyen funciones para entrenamiento como MATLAB. Este tipo de programas nos permiten enfocarnos en el proceso de entrenamiento más que en la implementación del mismo.

2.7 Redes Neuronales con Retraso en el Tiempo.

Otra topología empleada en los sistemas de reconocimiento de voz es la Red Neuronal con Retraso en el tiempo o TDNN (Time Delay Neural Network), en esta arquitectura se toman en cuenta las características dinámicas de la voz y es una variación del perceptrón multicapa expuesto anteriormente.

La red neuronal con retrazo en el tiempo fue propuesta por Lang y Hinton en 1988 y Waibel en 1989, es una red cuyas neuronas de las capas oculta y de salida son "replicadas" a lo largo del tiempo.

Para que esta red pueda ser considerada dinámica debe tener memoria [20]. Una forma de cumplir con esto es introducir retrasos en el tiempo (time delays) en la

estructura sináptica de la red, es decir las unidades básicas de este tipo de redes difieren un poco del modelo mostrado en la figura 2.2 como se explica en la siguiente sección.

- Modelo espacio temporal de la neurona.

En las redes TDNN el modelo de neurona de la figura 2.2 se modifica para tener en cuenta la naturaleza temporal de sus datos de entrada, esto se realiza a través de un filtro lineal invariante en el tiempo colocado entre la sinapsis de la neurona, tal como se muestra en la figura 2.11.

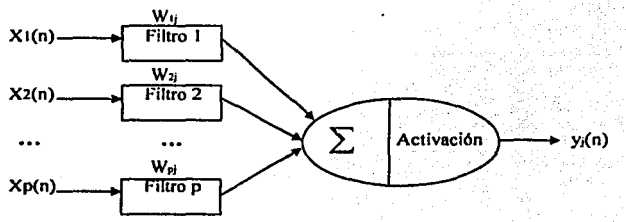


Figura 2.11. Modelo espacio temporal de la neurona, donde Xp son las entradas de la neurona, Wpj son los pesos de sus conexiones y los filtros, son los que representan la conexión.

En la siguiente figura se muestra una sola sinapsis y una sola entrada, la cual es retrasada en el tiempo.

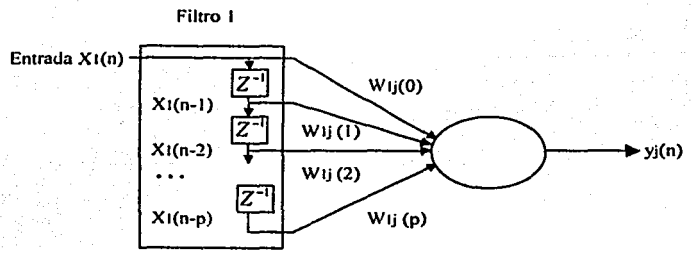


Figura 2.12. En la figura se muestra una sola conexión, la que existe entre la entrada X_1 y la neurona j-ésima, dicha entrada es retrasada por medio de Z^{-1}.

El comportamiento temporal de la sinapsis de la neurona *j*-ésima es descrito por la respuesta a impulso $h_{ji}(t)$ la cual actúa como la memoria de la sinapsis.

La respuesta de la sinapsis *i-ésima* es igual a la convolución entre su respuesta a impulso y la entrada $x_i(t)$. es decir:

$$h_{ij}(t) * x_i(t) = \int_{-\infty}^{t} h_{ij}(\lambda)\, x_i(t-\lambda)\, d\lambda \qquad 2.6$$

La actividad de la neurona debida a todas sus sinapsis esta dada por:

$$v_i(t) = \sum_{i=1}^{p} h_{ij}(t) * x_i(t) \qquad 2.7$$

$$= \sum_{i=1}^{p} \int_{-\infty}^{t} h_{ij}(\lambda)\, x_i(t-\lambda)\, d\lambda \qquad 2.8$$

Cada filtro de la sinapsis tiene las siguientes características [20]:

1. El filtro es causal, lo que significa que la sinapsis no responde antes de aplicar un estimulo a su entrada. Esto implica que la respuesta a impulso $h_{ij}(t)$ sea nula para valores de tiempo negativos:

$$h_{ij}(t) = 0 \quad \text{para } t < 0$$

2. Como se dijo, el filtro de la sinapsis representa su memoria siendo esta memoria finita, es decir:

$$h_{ij}(t) = 0 \quad \text{para } t > T$$

donde el valor de T es igual para todas las sinapsis y representa la duración de la memoria.

Tomando en cuenta las características anteriores, la ecuación 2.8 se puede escribir como:

$$v_i(t) = \sum_{i=1}^{p} \int_{0}^{T} h_{ij}(\lambda)\, x_i(t-\lambda)\, d\lambda \qquad 2.9$$

La convolución de la ecuación anterior se puede aproximar por medio de una sumatoria:

$$v_i(n) = \sum_{i=1}^{p} \sum_{l=0}^{M} w_{ij}(l)\, x_i(n-l) \qquad 2.10$$

donde:

$M = \frac{T}{\Delta t}$ es el número total de retrasos

Δt es el periodo de muestreo y

$w_{ij}(l) = h_{ij}(l)$ representa lo pesos de conexión entre la neurona i y la j.

Para representar en una forma más simplificada a la ecuación 2.10, esta se representa en forma matricial mediante la ecuación:

$$v_i(n) = \sum_{j=1}^{p} [w_{ij}][x_i(n)] \qquad 2.11$$

donde:

$$x_i(n) = [x_i(n), x_i(n-1), \ldots . x_i(n-M)]^T \quad \text{y}$$
$$w_{ij} = [w_{ij}(0), w_{ij}(1), \ldots, w_{ij}(M)]$$

Como en el caso de la neurona de la figura 2.2, para obtener la salida total de la neurona de una red con retraso en el tiempo, se debe utilizar una función de activación, si esta función es una sigmoide logística la salida esta dada por:

$$y_i(n) = \frac{1}{1+e^{-(v_i(n))}} \qquad 2.12$$

Si en lugar de utilizar matrices en la ecuación 2.11, empleamos valores escalares, la salida de una neurona de la red con retraso en el tiempo, se convierte en la salida de una neurona del perceptrón multicapa dada por la ecuación 2.2.

Utilizando el modelo dinámico de la neurona mostrada en la figura 2.11, podemos construir redes con retraso en el tiempo utilizando la topología de un perceptrón multicapa, en donde la única diferencia consiste en obtener las salidas de las neuronas por medio de la ecuación 2.12

Para ajustar los pesos de las conexiones de la red, se utiliza un algoritmo de entrenamiento supervisado, en el cual la respuesta actual de cada neurona es comparado con la respuesta deseada en cada instante de tiempo. Uno de los métodos de entrenamiento para las redes neuronales con retraso en el tiempo es el algoritmo de retropropagación a través del tiempo.

- Algoritmo de retropropagación a través del tiempo.

El algoritmo de Retropropagación a Través del Tiempo o "Backpropagation Through Time", es una extensión del algoritmo de retropropagación estándar que se utiliza en el perceptrón multicapa.

La diferencia es que los datos de entrenamiento son mostrados a la red en épocas. Si denotamos el tiempo de inicio de cada época por n_0 y su final por n_1, la función de error que se debe minimizar esta dado por :

$$E_{TOTAL}(n_0, n_1) = \frac{1}{2} \sum_{n=n_0}^{n_1} \sum_{j} e_j^2(n) \qquad 2.13$$

donde el subíndice j representa las neuronas de la capa de salida, la sumatoria $e_j(n)$ es la señal de error a la salida de cada neurona, medida con respecto a la salida deseada.

Al igual que el algoritmo de retropropagación estándar, este se divide en dos fases, la retropropagación hacia delante y hacia atrás.

En la fase de retropropagación hacia delante se calculan las salidas de las neuronas de cada capa empleando la ecuación 2.12. Los pesos de conexión, al igual que los datos de entrada y las salidas deseadas se guardan y son utilizadas en la siguiente fase.

En la fase de retropropagación hacia atrás, se calculan los valores del gradiente de error por medio de:

$$\delta_i(n) = \begin{cases} \varphi'(v_i(n))\, e_i(n) & si\ \ n = n_1 \\ \varphi'(v_i(n))\left[e_i(n) + \sum w_i \delta(n+1)\right] & si\ \ n_0 < n < n_1 \end{cases} \qquad 2.14$$

donde φ' es la derivada de la función de activación con respecto a sus argumentos.

Los valores de gradiente se calculan en sentido inverso, es decir comenzando en el tiempo n_1 y terminando en el tiempo n_0.

Finalmente, los pesos de las conexiones w_{ij} de la neurona i-ésima son ajustados por medio de la siguiente expresión:

$$\Delta w_{ij} = -\eta \frac{\partial E_{TOTAL}(n_0, n_1)}{\partial w_{ij}} \qquad 2.15a$$

$$= \eta \sum_{n=n_0+1}^{n_1} \delta(n)\, x_i(n-1) \qquad 2.15b$$

donde η es el factor de aprendizaje y $x_i(n-1)$ es la entrada i-ésima de la neurona en el tiempo $n-1$.

Aún cuando el algoritmo de retropropagación a través del tiempo es una extensión del algoritmo estándar, su implementación es más complicada, por esta razón se decidió entrenar las redes con retraso en el tiempo con el programa NICO, el cual tiene implementado dicho algoritmo.

2.8 Errores en el entrenamiento

Cuando se termina con el proceso de entrenamiento, es decir cuando el error entre la salida que nos proporciona la red y la salida real se aproxima lo suficiente al valor de error deseado, se debe probar su desempeño por medio de una validación cruzada, es decir comparando el error obtenido con los patrones de entrenamiento y el error que resulta de introducir a la red patrones de prueba que no hayan sido empleados en el entrenamiento. El error generado por los patrones de entrenamiento se denomina error en aprendizaje, mientras que el obtenido con los patrones de prueba se llama error de

generalización. Idealmente una arquitectura de red debe entrenarse hasta alcanzar el punto óptimo en el que el error de generalización es mínimo. [14]

Si se entrena una red empleando un error muy pequeño en la fase de aprendizaje, el error de generalización aumenta. Esto se debe a que al principio, la red se adapta progresivamente, sin embrago, en un momento dado el sistema se ajusta demasiado a las particularidades de entrenamiento, aprendiendo incluso el ruido presente en ellos, por lo que al presentarle patrones diferentes esta no reconoce algunos datos o en el peor de los casos no reconoce ninguno. Se dice entonces que la red ha "memorizado" los patrones del conjunto de entrenamiento, produciendo un fenómeno que se conoce como sobreaprendizaje.

Dado que el algoritmo de retropropagación es una generalización del método de descenso de gradiente, una red sigue el contorno de una superficie de error con los pesos moviéndose en dirección descendente.

Curva de la función de error

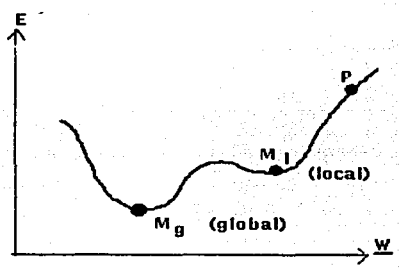


Figura 2.13. Gráfica de la función de error en la que se aprecia el mínimo local y el mínimo global. El entrenamiento puede oscilar alrededor del mínimo local y nunca alcanzar el global.

Para Perceptrones de dos capas (una de entrada y una de salida) la curva de error es cóncava y el método de descenso por gradiente siempre encuentra el mínimo error que tiene la función (parte más baja de la curva). Este error se conoce como mínimo global. Sin embargo cuando se agregan capas ocultas surge el problema de que la superficie de error contenga múltiples mínimos locales y dado que algunos mínimos están más abajo que otros, como se aprecia en la figura 2.13 (Ml más abajo que P), es posible que el método de descenso de gradiente no encuentre el mínimo global, provocando que el proceso de aprendizaje quede estancado en uno de estos mínimos locales. [14]

Para lograr que la red haga una buena generalización del los datos de entrada, es necesario entrenarla con los datos más representativos del problema a tratar, evitando al

máximo información redundante. En el capitulo III se explican algunos métodos empleados en los sistemas de reconocimiento de voz para este fin.

CAPITULO III

MÉTODOS PARA LA OBTENCIÓN DE LAS CARACTERÍSTICAS DE LAS SEÑALES DE VOZ

MÉTODOS PARA LA OBTENCIÓN DE LAS CARACTERÍSTICAS DE LAS SEÑALES DE VOZ.

Como se dijo en el capitulo I, el principal problema que se debe solucionar para lograr que un sistema de reconocimiento proporcione respuestas correctas a la información que recibe, es la disminución del número de datos de entrada. En el caso del reconocimiento de voz los datos de entrada al sistema son palabras, para reducir la cantidad de información contenida en estas señales de voz, es necesario procesarlas con algún método que nos permita obtener las características propias de cada palabra y no las características de cada individuo.

Existen varios métodos para extraer las características de la voz, dentro de los cuales se encuentran la Transformada Rápida de Fourier (FFT), los coeficientes de predicción lineal (LPC) y los coeficientes cepstrum en escala mel (MFCC). Todos estos métodos permiten extraer las características espectrales de la voz, esto significa que los valores que nos proporcionan son una representación de cómo se distribuyen las frecuencias que constituyen dicha señal.

Cuando se separan las componentes en frecuencia de una señal se obtiene un conjunto de valores que reciben el nombre de espectro en frecuencia. Aún cuando la información que nos proporciona el espectro en frecuencia es útil para muchas aplicaciones, para el desarrollo de un sistemas de reconocimiento de voz esta información no es la adecuada, dado que los espectros para una misma palabra son diferentes. En su lugar se emplea una versión "suavizada" del espectro, la cual se obtiene por medio de los coeficientes de predicción lineal o los coeficientes cepstrum. Una versión suavizada del espectro significa que no se toma en cuenta información detallada de la distribución de frecuencias de cada individuo, esto ayuda a las redes neuronales a realizar una mejor generalización de los datos de entrenamiento.

Tanto los sistemas de reconocimiento de voz como el oído humano reciben como datos de entrada sonidos provenientes del medio en el que se encuentran. Estos sonidos deben ser procesados para que puedan ser interpretados.

Un sonido es una compleja serie de cambios de presión en el aire generados por alguna fuente. En el caso de la voz, los sonidos son producido por exhalaciones de aire proveniente de los pulmones; este aire pasa por la laringe (también conocida como manzana de Adán) y sale por la boca o nariz. La laringe contiene músculos llamados cuerdas vocales los cuales alteran el aire que pasa a través de ellos para generar consonantes o vocales. Si las cuerdas están muy juntas, pero no completamente cerradas, estas vibran al pasar el aire a través de ellas produciendo sonidos vocales y algunas consonantes como la b, d, g y z; por otra parte si las cuerdas están separadas estas no vibran y producen sonidos secos como las consonantes p, t, k y f.

El área arriba de las cuerdas vocales se llama tracto vocal y este se encuentra dividido en dos: el tracto oral compuesto por la lengua, los labios, los dientes y el paladar y el tracto nasal con el cual se producen sonidos nasales.

En pocas palabras para que se produzca la voz es necesario que el tracto vocal sea excitado por el aire proveniente de los pulmones.

Un modelo simple pero efectivo del proceso de producción de voz es el que toma en cuenta tanto al tracto vocal como a su excitación. Es decir un modelo que contemple tanto los órganos involucrados en su generación, como el tipo de excitación que estos reciben.

Una forma de representar al tracto vocal, así como las excitaciones a los que esta sujeto se muestra en el diagrama de la figura 3.1 En donde H(n) es la función de transferencia del filtro digital variante en el tiempo que representa el tracto vocal, $s(n)$ es la salida final del sistema que representa la señal de voz, $u(n)$ es una fuente de excitación unitaria y G es la ganancia aplicada a dicha excitación.

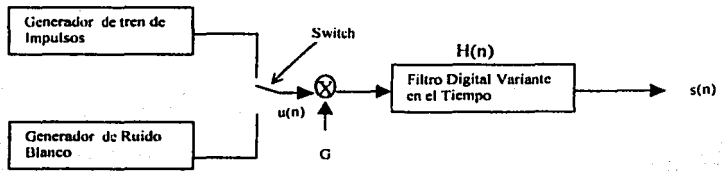


Figura 3.1. Modelo del tracto vocal y su excitación. El filtro digital representa el tracto vocal, mientras que el generador de tren de impulsos y el generador de ruido blanco representan las excitaciones que recibe.

Las fuentes de excitación del filtro digital son dos: un tren de pulsos cuasi-periódicos que representa la vibración de las cuerdas vocales o una fuente de ruido blanco que representa el flujo de aire que pasa a través de las cuerdas sin producir vibración. [10] La fuente de excitación se elige por medio de un switch cuya posición controla el carácter vibración /no vibración de la voz.

Con este modelo la configuración del tracto vocal puede ser estimada identificando el efecto de filtrado que se realiza sobre la excitación.

Para poder transmitir la voz o procesarla por medios electrónicos, es necesario convertir las variaciones de presión que se emiten cada vez que pronunciamos palabras, en valores eléctricos que puedan ser manipulados por algún instrumento. El transductor más utilizado para este fin es el micrófono, el cual convierte la voz en señales eléctricas continuas en el tiempo.

Si graficamos la señal de voz en un eje coordenado de dos dimensiones obtendremos una representación similar a la mostrada en la figura 3.2, en la cual la variable independiente es el tiempo y la dependiente es la amplitud.

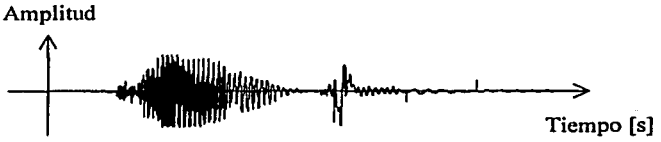


Figura 3.2. Señal de voz graficada con respecto al tiempo.

Dos características importantes de una señal, son su frecuencia y su amplitud.

La frecuencia es el número de veces que una onda de la señal se repite. La unidad utilizada para medir la frecuencia es el Hertz y se representa por medio de las letras Hz. El ser humano es capaz de escuchar sonidos dentro un rango de frecuencias comprendido entre los veinte y los veinte mil hertz.

La amplitud, en el caso de las señales de voz indica como varía la presión del aire en un instante de tiempo dado. Un valor grande de amplitud significaría que el micrófono captó un cambio de presión mayor al atmosférico, mientras que un valor de amplitud negativo significa que el cambio de presión fue menor.

Por mucho tiempo el procesamiento de voz se hacia sobre señales analógicas, es decir se debía trabajar sobre señales continuas en el tiempo las cuales tienen un número infinito de puntos a lo largo de su duración. Esto provocaba que el procesamiento fuera tardado para algunas aplicaciones y que además se perdiera calidad en la señal. Para un mejor manejo de la señal esta se digitaliza, es decir se toman solo algunas muestras de la señal continua. Este procedimiento se denomina muestreo y se realiza a través de un convertidor analógico digital. Para los fines de esta tesis, dicho convertidor se encuentra en la tarjeta de sonido de la computadora en la cual se capturan las señales de voz.

El intervalo de tiempo en el cual se toma una muestra después de otra se conoce como tiempo de muestreo T_s pero se utiliza más su valor inverso f_s llamado frecuencia de muestreo, es decir $f_s = 1/T_s$.

La señal digital contiene la misma información que la señal continua en el tiempo si la frecuencia de muestreo es al menos el doble de la frecuencia más alta contenida en la señal continua. Es decir $f_s = 2f$, a esto se le conoce como el teorema de Nyquist.

Para sistemas que transmiten y procesan la voz, como el teléfono por ejemplo, las frecuencias comúnmente utilizadas son 8,000 y 16,000 Hz.

¿ Por qué el procesamiento digital es preferible sobre el analógico?

Porque además de que se puede mejorar la calidad del sonido con el procesamiento digital, mucha teoría del procesamiento digital de señales ha sido diseñada para realizar

sofisticados procesos que son fáciles de implementar en computadoras o procesadores, esto permite consumir menos tiempo de cómputo y además las técnicas digitales son menos costosas que su contraparte analógica.

Una vez que se ha digitalizado la señal de voz, esta debe ser procesada para obtener de ella los valores que serán introducidos a la red. El procesamiento aplicado a las señales de voz dependerá del método empleado para la obtención de sus características, sin embargo para los dos métodos utilizados en esta tesis existen pasos comunes, estos son: preénfasis, segmentación de la señal y ventaneo.

3.1 Preénfasis

Las características del tracto vocal se obtienen en el dominio de la frecuencia por medio de la localización de las frecuencias correspondientes a sus resonancias. Estas corresponden a valores grandes en amplitud.

Los hombres hablan en tonos graves, es decir emiten señales de voz compuestas en su mayoría por frecuencias bajas, mientas que la voz de mujeres y niños se componen de frecuencias altas, correspondientes a tonos agudos. En general, las componentes espectrales de las frecuencias altas de la voz tienen menor amplitud con respecto a las que se encuentran en frecuencias más bajas, este fenómeno se acentúa cuando el locutor es un niño o una mujer. [3]

Si se desea que nuestro sistema sea capaz de reconocer la voz de cualquier persona, sea esta un hombre, una mujer o un niño debemos incluir un procesamiento que evite perder información contenida en las frecuencias altas, para ello se emplea un filtro que nos permita obtener una amplitud similar para todas las frecuencias. Este tipo de procesamiento recibe el nombre de preénfasis.

El proceso de preénfasis consiste pues, en pasar la señal de voz a través de un filtro digital paso altas de primer orden que amplifique las componentes de frecuencia alta para obtener una amplitud similar para todas las frecuencias. Este proceso es importante puesto que incluirá toda la información de la señal, aún la que se encuentra en las frecuencias altas. El filtro utilizado tiene como función de transferencia:

$$H(z) = 1 - \frac{a}{Z} \qquad 3.1$$

donde a es una constante que recibe como nombre *parámetro de préenfasis* y se encuentra dentro del rango: $0.9 \leq a \leq 1$.

Un valor típico de a empleado en los sistemas de reconocimiento de voz es de 0.9375 (15/16), con este valor las componentes de frecuencia alta se amplifican en más de 20 dB. [3]

La salida del filtro se relaciona con la señal de entrada por medio de la ecuación en diferencias:

$$s'(n) = s(n) - a\,s(n-1) \qquad 3.2$$

En la figura 3.3 se aprecia el efecto que provoca el filtrado sobre la señal original. En el espectro de la derecha se puede observar que las frecuencias altas son amplificadas evitando con ello perdida de información.

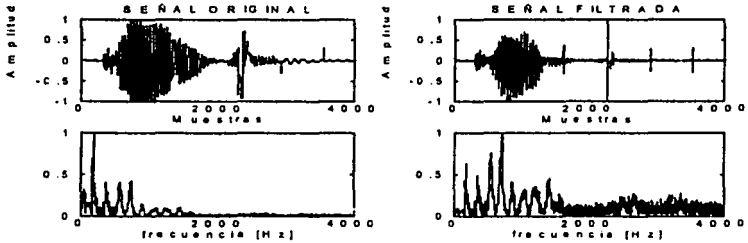


Figura 3.3. Efecto que se obtiene al aplicar a la señal de voz un preénfasis. En las gráficas de la izquierda se muestran la señal antes del preénfasis y su respectivo espectro, en la gráfica de la derecha se observa el efecto después del procesamiento.

3.2 Segmentación

El método de los coeficientes de predicción lineal, así como el método de los coeficientes cepstrum se aplican sobre señales estacionarias, es decir aquellas cuyas características no cambian con el tiempo.

La voz no es una señal estacionaria puesto que esta cambia constantemente con el tiempo y solo puede ser considerada como tal en pequeños intervalos. [10] Para obtener esos intervalos de tiempo la señal se debe dividir en pequeños segmentos llamados bloques en los cuales la señal se puede considerar como estática.

Cuando la señal de voz se divide en bloques es posible que se pierda información, para evitar esto cada segmento es traslapado, es decir se almacena información repetida la cual es distribuida en bloques diferentes. Esta redundancia de información es eliminada posteriormente.

La segmentación consiste en dividir la señal $s'(n)$ que resulta del proceso de preénfasis en bloques de N muestras. El primer bloque abarcará los primeros datos de la señal, el segundo bloque contendrá las últimas M muestras del primer bloque y N-M muestras nuevas, este proceso se repite hasta que todas las muestras de la señal se encuentran en algún bloque.

El valor de M depende del número de muestras que se desea tenga el traslape, en esta tesis dicho valor se tomó como un medio del valor de N, es decir: M = N/2. Con este valor se tiene que: el primer bloque consta de N muestras, el segundo bloque comienza M muestras después del primer bloque y se traslapa con este igual número de muestras. De forma similar, el tercer bloque comienza 2M muestras después del primer bloque (o M muestras después del segundo) y esta traslapado con el segundo bloque por M muestras, este procedimiento continúa hasta que toda la señal se encuentra distribuida dentro de los bloques.

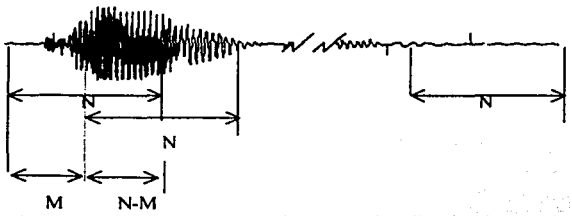


Figura 3.4. Proceso de segmentación: la señal se divide en bloques de N muestras con un traslape entre bloques igual a N-M muestras.

Cuando una señal es dividida en bloques se dice también que esta es segmentada de forma rectangular empleando la función:

$$w(n) = \begin{cases} 1 & \text{si } 0 \le n \le N-1 \\ 0 & \text{en cualquier otro caso.} \end{cases} \qquad 3.3$$

Como el proceso de segmentación es en realidad un ventaneo muchos autores manejan la segmentación y el traslape como un solo proceso denominado ventaneo con traslape.

El valor de N esta ligado a la resolución del espectro en frecuencia de la siguiente forma:

Denotando la resolución del espectro en frecuencia como Δf y tomando en cuenta que la frecuencia normalizada es igual a:

$$\omega = \frac{2\pi f}{f_s} \qquad 3.4$$

despejando f de la ecuación 3.4 y substituyéndolo por Δf, se tiene:

$$\Delta f = \frac{\Delta \omega f_s}{2\pi} \qquad 3.5$$

Si muestreamos el espectro en frecuencia de forma uniforme en el eje ω:

$$\Delta \omega = \frac{2\pi}{N} \qquad 3.6$$

sustituyendo 3.6 en 3.5 tenemos que:

$$\Delta f = \frac{\Delta \omega f_s}{2\pi} = \frac{f_s}{N} \qquad 3.7$$

Por lo tanto, para encontrar el número de muestras que tendrá cada ventana, se despeja esta variable de la ecuación anterior:

$$N = \frac{f_s}{\Delta f} = \Delta t \cdot f_s \qquad 3.8$$

donde el valor de Δt es el tamaño de la ventana.

Usualmente el tamaño de ventana empleado en los sistemas de reconocimiento de voz es de 20-40 ms, si se utiliza un tiempo menor se tiene un impacto muy grande en la resolución de la frecuencia, provocando que no se estime correctamente la posición de las frecuencias fundamentales del tracto vocal [3]. Por el contrario si se incrementa la resolución, es decir si se hace más grande la ventana, se puede identificar bien cada armónica del espectro, pero esto contrasta con la necesidad de dividir la señal en pequeños segmentos cuasi estáticos.

3.3 Ventaneo.

Como se dijo anteriormente, cuando la señal es dividida en bloques se realiza un ventaneo aplicando la función 3.3. El problema con este tipo de ventanas es que se producen discontinuidades o cambios bruscos cuando la señal pasa de un valor de cero a un valor máximo, esto provoca una distorsión en el espectro en frecuencia de la señal. Para prevenir este tipo de distorsiones debemos utilizar una ventana que minimice las discontinuidades de la señal al principio y final de cada bloque.

Una ventana empleada en los sistemas de reconocimiento de voz, con las características anteriores, es la ventana de Hamming, definida por la siguiente función:

$$w(n) = \begin{cases} 0.54 - 0.46 \cdot \cos\left(\dfrac{2\pi \cdot n}{N-1}\right) & n = 0, \ldots, N\text{-}1 \\ 0 & \text{para los demás valores} \end{cases} \qquad 3.9$$

donde n es el número de muestra que se está calculando y N es el número de muestras que tiene la ventana.

Para obtener los coeficientes LPC o MFCC, los bloques ventaneados y traslapados son procesados individualmente.

3.4 Método de los coeficientes de predicción lineal o LPC.

Uno de los métodos utilizados para obtener las características de la voz es el método de los coeficientes de predicción lineal o LPC (Linear Prediction Coefficients), dicho método se basa en la función de transferencia del modelo del tracto vocal de la figura 3.1.

Como se sabe, la función de transferencia de un sistema lineal se puede representar por medio de sus polos y ceros. Los ceros son el numerador de la función de transferencia y los polos son su denominador. En el caso de la función de transferencia que modela el tracto vocal de la figura 3.1, esta solo tiene ceros. [21]

Se ha demostrado de forma experimental que existe una relación entre las muestras adyacentes de las señales de voz. [7] Tomando en cuenta este hecho, si tenemos una señal de voz que fue muestreada en el tiempo s(n), está señal puede aproximarse mediante una combinación lineal de sus muestras pasadas p, es decir:

$$s(n) \approx a_1 s(n-1) + a_2 s(n-2) + \ldots + a_p s(n-p) \qquad 3.10$$

En forma más simplificada e incluyendo la excitación del tracto vocal $u(n)$, la ecuación 3.10 se puede representar de la siguiente forma:

$$s(n) = \sum_{k=1}^{p} a_k s(n-k) + G\,u(n) = \hat{s}(n) + G\,u(n) \qquad 3.11$$

en donde los valores a_k son los coeficientes de predicción lineal tomados del filtro digital del modelo de la figura 3.1 y G es la ganancia que se da a la excitación.

Si consideramos la combinación lineal de las muestras pasadas $\hat{s}(n)$ como un estimador de la señal $s(n)$, entonces, para saber que tan buena es nuestra predicción empleamos un parámetro de comparación definido como:

$$e(n) = s(n) - \hat{s}(n) = s(n) - \sum_{k=1}^{p} a_k s(n-k) \qquad 3.12$$

al cual se le conoce como error de predicción de la señal.

Como queremos que nuestras predicciones se acerquen a los valores reales de la señal, se debe encontrar el conjunto de coeficientes que minimicen el error de predicción. Para ello, en lugar de utilizar directamente la ecuación 3.12, minimizamos el error de predicción medio cuadrático dado por la ecuación:

$$e(n) = \sum \left(s(n) - \sum_{k=1}^{p} a_k s(n-k) \right)^2 \qquad 3.13$$

El método de los coeficientes de predicción lineal ó LPC consiste pues, en encontrar los coeficientes del filtro digital que minimizan el error dado por la ecuación 3.13, de tal forma que las propiedades espectrales del filtro digital correspondan a la palabra pronunciada.

Hay dos formas de encontrar los coeficientes de predicción lineal, una es por medio de la covarianza y la otra por medio de la autocorrelación. En general, para los sistemas de reconocimiento de voz se utilizan los coeficientes de predicción por medio de la autocorrelación.

- Autocorrelación.

En el método de la autocorrelación ó método de Durbin [23] la señal se autocorrelaciona para minimizar el error de predicción medio cuadrático dado por la expresión 3.13.

La función de autocorrelación nos permite tener una relación uno a uno con los coeficientes de predicción. Es decir los coeficientes de predicción se pueden calcular a partir de la función de autocorrelación.

Si asumimos que la señal $s(n)$ es cero para $n < 0$ y $n > N$ (lo cual se logra multiplicándola por una ventana), el error de predicción medio cuadrático se convierte en la expresión:

$$\sum_{k=1}^{p} a_k \cdot R_{|i-k|} = -R_i \qquad 1 \le i \le p \qquad 3.14$$

donde R_i es la función de autocorrelación de la señal $s(n)$ dada por:

$$R_i = \sum_{n=0}^{N-1-i} s(n) \cdot s(n+|i|) \qquad 3.15$$

Para obtener los coeficientes de predicción a_k la ecuación 3.14 se descompone de la siguiente forma:

$$E_0 = R_0 \qquad 3.16a$$

$$k_i = -\frac{\left(R_i + \sum_{j=0}^{i-1} a_j^{(i-1)} R_{i-j}\right)}{E_{i-1}} \qquad 3.16b$$

$$a_i^{(i)} = k_i \qquad 3.16c$$

$$a_j^{(i)} = a_j^{(i-1)} + k_i \cdot a_{i-j}^{(i-1)} \quad , \quad 1 \le j \le i-1 \qquad 3.16d$$

$$E_i = (1 - k_i^2) E_{i-1} \qquad 3.16e$$

El conjunto de ecuaciones 3.16b a 3.16e se resuelve recursivamente para $i = 1, 2, \ldots, p$ y los coeficientes de predicción lineal están dados por:

$$a_j = a_j^{(p)} \qquad 1 \le j \le p$$

El valor de R_0 es generalmente descartado y p es el número de coeficientes que se desean obtener.

Típicamente el número de coeficientes utilizados en el reconocimiento de voz va de los 8 a los 16. Con esta cantidad de coeficientes, el número de datos de la señal se reduce aproximadamente a un décimo de su tamaño original.

Los LPC son un buen modelo de la señal de voz [10]. Sin embargo esto es cierto solo si la señal es estática.

Dado que las características espectrales de la voz varían con el tiempo, la señal debe ser ventaneada para obtener pequeños segmentos en los cuales sea estática y en cada uno de ellos encontrar los coeficientes de predicción lineal.

El diagrama de bloques de la figura 3.5 muestra los pasos para obtener los coeficientes LPC.

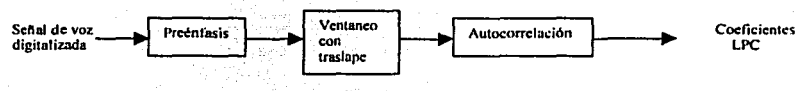


Figura 3.5. Diagrama de bloques del método de los coeficientes de predicción lineal (LPC).

A continuación se explica la forma en la cual se obtienen los coeficientes de predicción lineal tomando como base el diagrama de bloques de la figura anterior.

El primer paso en el método consiste en dar a la señal de voz una amplitud similar para todas las frecuencias, para tal efecto se emplea la ecuación 3.2 con la cual se amplifican las altas frecuencias, evitando con ello perdida de información.

Si representamos a la señal de voz digitalizada por medio de $s(n)$, a la señal después del preénfasis por $s'(n)$ y elegimos a = 0.9375 como valor de preénfasis, la salida de este bloque esta dada por:

$$s'(n) = s(n) - 0.9375\, s(n-1)$$

donde n representa el número de muestras de la señal digital. Este valor dependerá del tamaño de la señal de voz.

Después del preénfasis la señal s'(n) es dividida en bloques de N muestras con un traslape entre bloques igual a N-M. Para determinar el número de muestras que tendrá cada bloque se multiplica la frecuencia de muestreo f_s por la duración de la ventana, esto es:

$$\textit{Número de muestras por ventana} = (f_s)\ (\textit{segundos por ventana})$$

Por ejemplo si se utiliza un tamaño de ventana igual a 90 ms y una frecuencia de muestreo de 8 kHz, el número de muestras por ventana es igual a 720. Como el valor de M (número de muestras traslapada) se toma como ½ del valor de N, cada ventana es traslapada en 360 valores.

Para determinar en cuantas ventanas sin traslape cabe la señal, se divide el número de muestras que componen la señal $s'(n)$ entre el número de muestras por ventana, esto es:

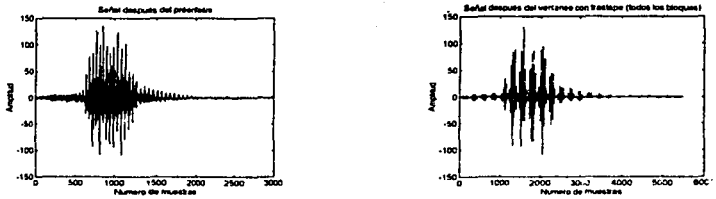
$$\textit{Número de Ventanas sin traslape} = \frac{\textit{muestras de la señal } s'(n)}{\textit{Número de muestras por ventana}}$$

El número de ventanas sin traslape es un valor entero, por lo cual, si el resultado de esta operación es un número fraccionario, este se debe redondear al entero mayor, porque si se eligiera un valor entero menor, se perdería parte de la información de la señal.

Dado que se utiliza un traslape igual a la mitad del número total de muestras en cada bloque, el número total de ventanas es igual a dos veces el número de ventanas sin traslape menos uno.

El siguiente paso del método consiste en multiplicar cada uno de los bloques por la ventana de Hamming dada por la ecuación 3.5.

Las señales después del preénfasis y del ventaneo con traslape se muestran en la figura 3.6. En el caso de la gráfica que muestra todos los segmentos, se tomó como 90ms el tamaño de cada ventana.



(a) (b)

Figura 3.6. En la gráfica de la izquierda se muestra la señal después del preénfasis, en la gráfica de la derecha se muestran todos los segmentos en los que fue dividida la señal (nueve segmentos en este caso) .

Para calcular la autocorrelación de cada ventana se utilizó la función *xcorr* del programa Matlab, la cual nos da una respuesta como la mostrada en la figura 3.7. Como se puede observar, esta gráfica es simétrica y para nuestro desarrollo solo es necesaria la parte positiva, reduciendo con ello el número de muestras a la mitad. El número de elementos que se toman en la autocorrelación, esta relacionado con el número de coeficientes de predicción lineal que se desean obtener.

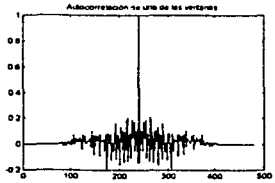


Figura 3.7 Autocorrelación de una de las ventanas.

Finalmente, para calcular los coeficientes de predicción lineal no se utilizaron directamente las ecuaciones 3.16, en su lugar se empleó la función *lpc* de Matlab. Dicha función nos proporciona coeficientes de predicción lineal entre 0 y 1. Se toman los trece

primeros coeficientes de predicción porque el primer predictor siempre es uno y es descartado. Obteniendo con ello solo doce predictores.

Se toman los coeficientes de predicción lineal de cada ventana y con ellos se forma un solo vector, cuyos valores serán tomados como los datos de entrada de la red.

El orden en el cual se agruparon los coeficientes de las diferentes ventanas para formar un solo vector fue el siguiente: se colocaron los doce coeficientes correspondientes a la primer ventana, después los coeficientes de la segunda ventana y así sucesivamente hasta tener un solo vector. En la siguiente figura se muestra el vector de coeficientes que representa a la palabra "fin".

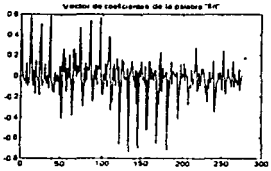


Figura 3.8. Coeficientes de predicción lineal correspondientes a la palabra "fin". Este vector se obtuvo al emplear un tamaño de ventana igual a 32 ms y tomando 12 coeficientes de predicción por ventana.

3.5 Método de los coeficientes cepstrum en escala Mel o MFCC.

En 1963 Bogert, Healy y Tukey desarrollaron el procesamiento cepstral para aplicarlo en el estudio de señales sísmicas. [16] Unos años más tarde Oppenheim combinó este procesamiento con un modelo matemático que se encontraba desarrollando llamado homomorfismo y llego a la conclusión de que los cepstrum se podían generalizar y ser empleados no solo en el análisis de señales sísmicas sino también en el análisis y síntesis de voz. [17]

La palabra cepstrum fue propuesta por Bogert y sus colaboradores para distinguir su método y es el anagrama inglés de la palabra espectro (spectrum), propusieron este nombre porque el cepstrum es el espectro de un espectro.

Matemáticamente el cepstrum se define como la transformada inversa del modulo del espectro en escala logarítmica o la transformada inversa del logaritmo del módulo de la transformada de la señal, esto es:

$$c(n) = F^{-1}[log\,|X(z)|] \qquad 3.17$$

en donde $c(n)$ en el n-ésimo coeficiente cepstrum, $X(z)$ es la trasformada de Fourier de la señal $x(t)$ y $F^{-1}[log\ |X(z)|\ ]$ es la transformada inversa de Fourier del logaritmo de la transformada $X(z)$.

Al igual que los coeficientes LPC, los coeficientes cepstrum empleados en los sistemas de reconocimiento de voz se basa en el modelo del tracto vocal mostrado en la figura 3.1.

Si representamos a la excitación en función del tiempo como $u(t)$ y la función de transferencia del tracto vocal como $h(t)$ la salida del sistema tracto vocal-excitación esta dada por:

$$s(t) = e(t) * h(t) \qquad 3.18$$

Aplicando la transformada de Fourier en ambos lados de la ecuación 3.18, obtenemos la salida del sistema en el dominio de la frecuencia:

$$S(z) = E(z)\,H(z) \qquad 3.19$$

Como se observa en la ecuación 3.19, la salida del sistema esta dada por una multiplicación entre la fuente de excitación y el filtro que representa el tracto vocal. Podemos separar la información que contiene cada miembro si se aplica en ambos lados de la ecuación un logaritmo:

$$log\,(\,S(z)\,) = log(\,E(z)\,) + log(\,H(z)\,) \qquad 3.20$$

Al aplicar el logaritmo a la ecuación 3.19, esta se transforma en una operación lineal permitiéndonos con ello realizar sobre ella operaciones lineales para separar las altas componentes cepstrales de las bajas. Dicha separación se lleva a cabo mediante un filtrado.

Las altas componentes cepstrales corresponden a variaciones rápidas en el espectro y se relacionan estrechamente con la frecuencia fundamental y el carácter periódico de la excitación aplicada al tracto vocal. [6]

Las bajas componentes cepstrales corresponden a variaciones lentas de las componentes espectrales y se relaciona con la respuesta en frecuencia del filtro que modela el tracto vocal.

La mayor parte de la información del locutor proviene de la excitación de las cuerdas vocales, haciendo estas características de excitación dependientes del locutor, por otro lado tenemos que las características del tracto vocal son las mismas sin importar que persona sea la que pronuncie las palabras; por lo tanto para reconocer palabras o letras y no locutores lo que necesitamos son las bajas componentes cepstrales.

Una vez separadas las bajas componentes cepstrales por medio de un filtro, los coeficientes cepstrum se obtienen calculando la transformada inversa de Fourier. Este tipo de cepstrum recibe el nombre de cepstrum real.

Además del cepstrum real existe el complejo, la única diferencia entre ambos es que en el cepstrum complejo se calcula la transformada inversa del logaritmo de la transformada sin calcular el módulo. El cepstrum complejo tiene la ventaja de ser invertible, es decir a partir de él es posible recuperar de nuevo la señal original, sin embargo es más difícil de calcular que el cepstrum real y para los fines de esta tesis no requerimos obtener de nuevo la señal original, sino una representación de la voz que podamos utilizar para entrenar a la red neuronal.

Son dos las características por las cuales los cepstrum son utilizados en el reconocimiento de voz:

- La primera es que los cepstrum suavizan fácilmente el espectro. Como se había dicho, una versión suavizada del espectro nos proporciona la información necesaria para el reconocimiento de palabras eliminado detalles de la señal de voz propias de cada locutor.

- La segunda es una variante de los coeficientes cepstrum llamada MFCC

En los coeficientes cepstrum reales en escala Mel o MFCC (Mel Frequency Cepstrum Coefficients) se toma en cuenta la forma en la cual el oído percibe los sonidos, esta consideración es muy útil puesto que nos permite tener una representación de las señales de voz que incluya tanto su emisión como su percepción.

El oído humano es capaz de realizar un procesamiento eficiente sobre los sonidos que capta del medio en el que se encuentra, es tan eficiente que puede distinguir sonidos diferentes sin importar que estos provengan de diferentes direcciones y en ambientes ruidosos. Al tener un modelo que tome en cuenta las características del oído se aumentan las posibilidades de que el reconocimiento de palabras que haga nuestro sistema se parezca al que realiza una persona.

El método de los coeficientes ceptrum en escala mel para obtener las características de las señales de voz consta de los pasos que se muestran en el siguiente diagrama de bloques:

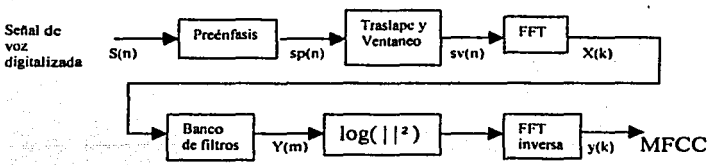


Figura 3.9. Diagrama de bloques del método de los coeficientes cepstrales en escala mel o MFCC.

Como se puede observar en el diagrama anterior el proceso para obtener los coeficientes MFCC tiene los primeros dos pasos del método de los coeficientes LPC: el preénfasis y el ventaneo con traslape, esto se debe a que ambos métodos realizan sus cálculos sobre intervalos de tiempo en los cuales la señal de voz se pueda considerar estática. El siguiente paso consiste en calcular para cada ventana el espectro en frecuencia por medio de la transformada de Fourier (FFT). Posteriormente este espectro se filtra por medio de un banco de filtros, se calcula su logaritmo y se aplica la transformada inversa de Fourier (FFT inversa). Estos tres pasos se describen a continuación con más detalle.

- 3.5.1 Banco de filtros

Análisis espectrales revelan que el oído realiza una discriminación de las frecuencias que componen los sonidos que llegan a él, dicha discriminación se lleva a cabo por medio de nervios ubicados en la membrana basilar que son sensibles a determinadas frecuencias. El comportamiento de estos nervios pude ser modelado por medio de un banco de filtros, en el cual cada filtro correspondería a un nervio en particular [3]. Generalmente se emplea un arreglo de 24 filtros.

Los veinticuatro filtros del banco están sintonizados a diferentes frecuencias a lo largo del ancho de banda de la señal analizada, en el caso de los coeficientes MFCC se emplea la escala mel. [6]

Un mel es una unidad que mide la forma en la cual el oído percibe la frecuencia natural de la voz. El termino mel lo propusieron Steven y Volkman en 1940 y es producto de experimentos realizados para determinar la relación que existe entre la frecuencia medida en hertz y la sensibilidad del oído a determinadas frecuencias. Steven y Volkman encontraron que el odio percibe las frecuencias de forma lineal hasta los 1000 Hz y después de este valor la percepción sigue un comportamiento logarítmico. Además se observó que la mayor cantidad de información se encuentra en los primeros 1000 Hz.

Para ilustrar el hecho de que existe una mayor cantidad de información dentro de los primeros 1000 Hz, se emplea la figura 3.10 en donde se grafica el espectro en frecuencia de la palabra "campo".

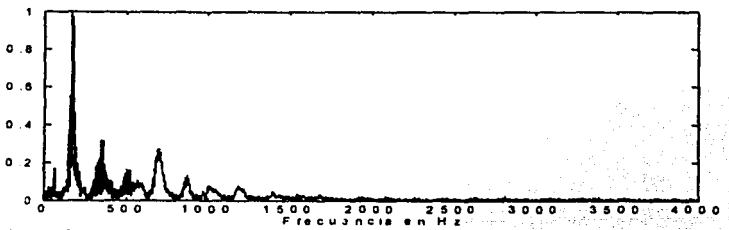


Figura 3.10. Espectro en frecuencia de la palabra "campo" en el cual se observa que la mayor información se concentra en los primeros 1000 Hz.

Como se puede observar en el diagrama anterior el proceso para obtener los coeficientes MFCC tiene los primeros dos pasos del método de los coeficientes LPC: el préenfasis y el ventaneo con traslape, esto se debe a que ambos métodos realizan sus cálculos sobre intervalos de tiempo en los cuales la señal de voz se pueda considerar estática. El siguiente paso consiste en calcular para cada ventana el espectro en frecuencia por medio de la transformada de Fourier (FFT). Posteriormente este espectro se filtra por medio de un banco de filtros, se calcula su logaritmo y se aplica la transformada inversa de Fourier (FFT inversa). Estos tres pasos se describen a continuación con más detalle.

- 3.5.1 Banco de filtros

Análisis espectrales revelan que el oído realiza una discriminación de las frecuencias que componen los sonidos que llegan a él, dicha discriminación se lleva a cabo por medio de nervios ubicados en la membrana basilar que son sensibles a determinadas frecuencias. El comportamiento de estos nervios pude ser modelado por medio de un banco de filtros, en el cual cada filtro correspondería a un nervio en particular [3]. Generalmente se emplea un arreglo de 24 filtros.

Los veinticuatro filtros del banco están sintonizados a diferentes frecuencias a lo largo del ancho de banda de la señal analizada, en el caso de los coeficientes MFCC se emplea la escala mel. [6]

Un mel es una unidad que mide la forma en la cual el oído percibe la frecuencia natural de la voz. El termino mel lo propusieron Steven y Volkman en 1940 y es producto de experimentos realizados para determinar la relación que existe entre la frecuencia medida en hertz y la sensibilidad del oído a determinadas frecuencias. Steven y Volkman encontraron que el odio percibe las frecuencias de forma lineal hasta los 1000 Hz y después de este valor la percepción sigue un comportamiento logarítmico. Además se observó que la mayor cantidad de información se encuentra en los primeros 1000 Hz.

Para ilustrar el hecho de que existe una mayor cantidad de información dentro de los primeros 1000 Hz, se emplea la figura 3.10 en donde se grafica el espectro en frecuencia de la palabra "campo".

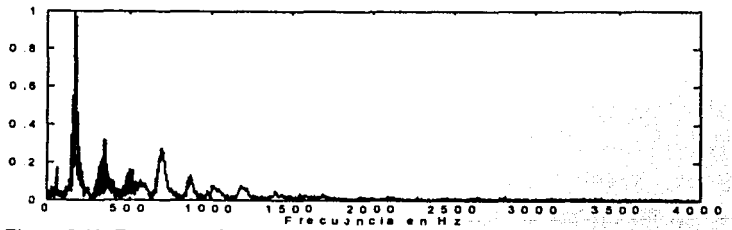


Figura 3.10. Espectro en frecuencia de la palabra "campo" en el cual se observa que la mayor información se concentra en los primeros 1000 Hz.

Siguiendo la escala mel, los 24 filtros que modelan la membrana basilar del oído se distribuyen de la siguiente forma: la frecuencia central de los primeros diez filtros es uniformemente espaciada antes de los primeros 1000 Hz para analizar la parte del espectro que contiene más información, después de 1 kHz la frecuencia central de los catorce filtros restantes siguen una distribución logarítmica.

La distribución de los filtros se muestra en la figura 3.11, en donde se observa también que el ancho de banda es el mismo para los primeros diez y aumenta para los filtros restantes.

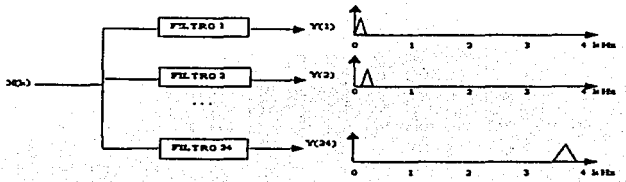


Figura 3.11. Distribución de los filtros que representan la membrana basilar del oído. La señal $X(k)$ es la trasformada de Fourier de cada ventana.

Con el objeto de facilitar el filtrado, este se realiza directamente en el dominio de la frecuencia multiplicando a la señal X(k) por una ventana triangular con ecuación:

$$T(k) = \begin{cases} |k| < \Delta_m \longrightarrow & 1 - \dfrac{|k|}{\Delta_m} \\ |k| > \Delta_m \longrightarrow & 0 \end{cases} \qquad 3.21$$

donde el valor de Δ_m es la mitad del ancho de banda del filtro.

Cada filtro del banco se caracteriza por una frecuencia central representada por b_m que determina la distribución de los filtros y un ancho de banda $2\Delta_m$ que indica el número de muestra que se tiene en cada filtro.

La salida de cada filtro esta dada por la ecuación:

$$Y(m) = \sum_{b_m-\Delta_m}^{b_m+\Delta_m} X(k) \cdot T(k - b_m) \qquad 3.22$$

en donde m indica el número de filtro que se esta calculando, T(k) es la función triangular definida anteriormente, b_m es la frecuencia central del filtro y $2\Delta_m$ es su ancho de banda.

El índice k se obtiene por medio de la relación entre la frecuencia ω de la señal discreta en el tiempo y la frecuencia f de su respectiva señal continua en el tiempo:

$$f = \frac{\omega f_s}{2\pi} \qquad 3.23$$

Si se realiza un muestreo uniformemente distribuido, es decir con el mismo intervalo de tiempo entre muestra y muestra, la frecuencia de la señal discreta esta dada por:

$$\omega = \frac{2\pi k}{N} \qquad 3.24$$

Substituyendo la ecuación 3.23 en la ecuación 3.24 y despejando se tiene que:

$$k = \frac{fN}{f_s} \qquad 3.25$$

Dado que los primeros diez filtros se distribuyen uniformemente la frecuencia central de cada uno de ellos debe ser proporcional, esto se logra por medio de la expresión:

$$b_m = b_{m-1} + \Delta_m \qquad 3.26$$

donde el valor de Δ_m es constante para este intervalo.

Para ubicar la frecuencia central de los filtros restantes se emplea también la ecuación 3.26, sin embargo, como la distribución de estos es logarítmica el valor de Δ_m deja de ser constante y se tiene que emplear la siguiente aproximación:

$$\Delta_m = 1.2 \cdot \Delta_{m-1} \qquad 3.27$$

Después de que se filtra el espectro por medio del banco, se obtiene una versión suavizada del mismo, realizando un proceso similar al que efectúa el oído humano.

- Cálculo del logaritmo de la energía.

El siguiente paso consiste en calcular el cuadrado de la magnitud de la señal obtenida con la ecuación 3.22 y aplicarle el logaritmo (se puede hacer una simplificación si se toma en cuenta la propiedad de los logaritmos, con la cual el logaritmo de un exponente se convierte en un factor de escala permitiendo calcular simplemente el logaritmo de la magnitud [3]). Este proceso se realiza también en el oído y tiene como finalidad descartar

información no relevante, dando como resultado que la extracción de características sea menos sensible a variaciones dinámicas. Matemáticamente esto sería:

$$log(\,|\,Y(m)|\,) \qquad 3.28$$

- 3.5.2 Cálculo de los coeficientes MFCC.

Finalmente, para obtener los coeficientes cepstrum en escala mel o MFCC se calcula la transformada inversa de Fourier de la señal obtenida en el bloque anterior, por medio de la siguiente ecuación:

$$y(k)=\sum_{m=1}^{M}\log\{|\,Y(m)\,|\}\cos\left(k\left(m-\frac{1}{2}\right)\frac{\pi}{M}\right)\quad k=0\ldots L \qquad 3.29$$

en donde i representa el número de coeficiente cepstrum que se esta calculando, $Y(m)$ es la salida del banco de filtros y M es el número total de filtros empleados, que en este caso es igual a 24.

El coeficiente de orden cero es equivalente al logaritmo de la energía de cada ventana y al igual que los LPC este valor generalmente se descarta debido a que es tomado en cuenta en la señal en el tiempo.

Típicamente el número de coeficientes cepstrales utilizados para los sistemas de reconocimiento voz es de 9 a 12. Al igual que los coeficientes de predicción lineal, esta cantidad de coeficientes permite reducir el número de muestras en aproximadamente un 10% de su tamaño original.

A continuación se explica la forma en la cual se obtienen los coeficientes MFCC tomando como base el diagrama de bloques de la figura 3.9.

Los primeros pasos del método de los coeficientes cepstrum en escala mel consisten en aplicar preénfasis a la señal de voz digitalizada y dividirla en bloques traslapados, es decir se utilizan los dos primeros pasos del método de los coeficientes de predicción lineal, por lo que no se explicaran de nuevo para centrarnos en la obtención de los coeficientes MFCC.

Después de aplicar a la señal de voz digitalizada un préenfasis y un ventaneo con traslape, se calcula la transformada rápida de Fourier de los datos de cada una de las ventanas resultantes, para tal efecto empleamos la función *fft* de Matlab. De este espectro se toma solo la parte positiva, a la cual nos referiremos como la señal *X(k)*.

La señal *X(k)* es filtrada por medio de un banco de veinticuatro filtros que simulan la forma en la cual el oído percibe los sonidos. Los diez primeros filtros son distribuidos uniformemente dentro de los primeros 1000 Hz para cubrir la zona de mayor información con más detalle. Dado que la frecuencia de muestreo utilizada en el presente trabajo fue de 8 kHz, los filtros restantes son distribuidos de forma exponencial de 1kHz a 4kHz.

El proceso de filtrado se lleva a cabo mediante la multiplicación de la señal *X(k)* (mitad del espectro) y la ventana triangular definida por la ecuación 3.21. Para ello empleamos la ecuación 3.22.

Los índices de la sumatoria de la ecuación 3.22 cambian dependiendo del filtro que se esta calculando y se obtienen de la siguiente manera:

Como los primeros diez filtros se distribuyen uniformemente dentro de los primeros 1000 Hz, se debe encontrar el número de muestras que corresponden a esta frecuencia es decir:

$$\text{Número de muestras en 1 KHz} = \left(\frac{1\text{ kHz}\ (\ \text{Número de muestras de X(k)}\ )}{\text{rango de frecuencia del espectro}} \right)$$

El número de muestras correspondientes a 1KHz se divide entre diez para obtener el valor de Δm. Este valor es constante para los primeros diez filtros y su frecuencia central esta dada por $b_m = b_{m-1} + \Delta_m$ (ecuación 3.26), donde el primer valor de *bm* es igual a Δm.

Para los catorce filtros restantes, también utilizamos la ecuación 3.26, pero el valor de Δm deja de ser constante y esta dado por $\Delta_m = 1.2 \cdot \Delta_{m-1}$ (ecuación 3.27)

Habiendo encontrado todos los valores de *bm* y Δm, se calcula el valor de cada filtro por medio de la ecuación 3.22 y se obtiene el conjunto de valores *Y(1), Y(2), ... ,Y(24)*. En la siguiente figura se muestra la salida del banco de filtros correspondientes a la primera ventana de la palabra "fin".

Figura 3.12. Salida del banco de veinticuatro filtros correspondiente a una sola ventana.

Finalmente, para obtener los coeficientes cepstrum en escala mel, hacemos uso de la expresión 3.29, la cual se rescribe a continuación:

$$y(k)=\sum_{m=1}^{24}\log\{|Y(m)|\}\ \cos\left(k\left(m-\frac{1}{2}\right)\frac{\pi}{24}\right)$$

Cada valor de *y(k)* es un coeficiente MFCC, para las pruebas siguientes se calcularon doce coeficientes (*y(1), y(2), ..., y(12)*) por ventana, los cuales fueron agrupados en un solo vector, tomando los doce coeficientes de la primer ventana como primeros elementos del vector y así sucesivamente. En la siguiente figura se gráfica el vector de coeficientes que representa a la palabra fin.

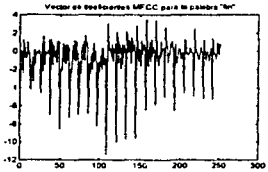


Figura 3.13. Vector de coeficientes MFCC de la palabra "fin". En este caso el tamaño de ventana utilizado fue igual a 32ms y el número de coeficientes ceptrum por cada ventana fue de 12.

Cabe recordar que tanto los coeficientes de predicción lineal obtenidos por medio del conjunto de ecuaciones 3.16 como lo coeficientes cepstrum en escala mel obtenidos por medio de la ecuación 3.29 corresponden a una sola ventana, por lo tanto para encontrar los coeficientes de toda la señal es necesario aplicar los métodos en todas y cada una de las ventanas, con lo cual se obtienen señales similares a las de las figuras 3.8 y 3.13.

CAPITULO IV

DESARROLLO

DESARROLLO.

Existen en el mercado muchas aplicaciones de sistemas de reconocimiento de voz, las cuales van desde programas de dictado que se encuentran al alcance de personas con medianos recursos económicos hasta aplicaciones más especializadas y caras como por ejemplo los sistemas que permiten el control del lente de un microscopio, los cuales solo pueden ser adquiridos por grandes instituciones.

Teniendo en cuenta este panorama, nuestro objetivo es pues establecer las bases para el desarrollo de un sistema de reconocimiento de voz barato, que pueda ser utilizado en aplicaciones sencillas y que además sea capaz de reconocer las palabras de su base en forma independiente del usuario. Para cumplir con nuestro objetivo, se realizaron pruebas con varias arquitecturas de redes neuronales artificiales utilizando estas como generalizadores de patrones.

Los patrones empleados como entrada del sistema se obtuvieron con los dos métodos descritos en el capitulo anterior, estos son: el método de los coeficientes de predicción lineal o coeficientes LPC y el método de los coeficientes cepstrum en escala mel o coeficientes MFCC.

Dichos patrones fueron entrenados con diferentes topologías de redes neuronales utilizando la arquitectura del perceptrón multicapa, así como redes neuronales con retraso en el tiempo, ambas descritas en el capitulo II.

Se probaron diferentes topologías y métodos de obtención de parámetros de voz con el fin de determinar que combinación de topología y método nos permitía reconocer un número mayor de patrones de entrada que no fueran presentados a la red con anterioridad.

Todo el desarrollo de las pruebas se hizo con una computadora personal con procesador Pentium MMX a 200Mhz. El software utilizado para grabar la voz de las personas empleadas en las fases de entrenamiento y prueba de la red neuronal fue el programa sndrec32 de Microsoft. Estas señales digitalizadas de voz eran guardadas y posteriormente procesadas ya sea con el método de los coeficientes LPC o el de los coeficientes MFCC.

El número de elementos que forman las señales digitalizadas depende de la frecuencia de muestreo, si se utiliza una frecuencia de 16 kHz se obtiene una señal digital con buena calidad de sonido, sin embargo a esta frecuencia la cantidad de muestras es grande provocando que el procesamiento consuma más tiempo de computo, así como espacio en memoria; por esta razón se decidió utilizar una frecuencia de 8 kHz, similar a la que se ocupa en la red telefónica y en la cual no se pierde demasiada información. Con ello el número de operaciones en los procesamientos aplicados a las señales y en los procesos de entrenamiento y prueba de la red se reduce, además de disminuir el espacio de almacenamiento necesario para guardar dichas señales.

Todas las señales de voz utilizadas a lo largo del desarrollo de esta tesis tienen las siguientes características:

Tipo de archivo:	ascii con formato *wav*
Frecuencia de muestreo (fs):	8 kHz
Precisión en la amplitud:	8 bits (valores enteros de –128 a 127)
Tipo de señal:	monoaural

Se grabaron las voces de ocho personas de las cuales tres son de mujeres con edades entre los veinte y sesenta años y las cinco voces restantes son de hombres con edades que van de los diecisiete hasta los setenta años, las palabras tuvieron que ser grabadas en diferentes días y a diferentes horas, pero se procuró evitar que en el momento de la grabación el ruido ambiental fuera excesivo y afectara la calidad de la señal de voz.

La habitación empleada para la grabación mide aproximadamente nueve metros cuadrados por dos y medio de alto.

La distancia de los locutores con respecto al micrófono varió en unos cuantos centímetros y se pidió a las personas que pronunciaran las palabras de forma natural, es decir utilizando el mismo tono de voz que tiene cuando platican con otra persona. Esto fue hecho con la finalidad de tener casos que representaran situaciones reales en las cuales se encontraría nuestro sistema.

Todas las señales de voz fueron acotadas, separando la parte que corresponde a la voz de aquella que solo es ruido o simplemente silencio. Para realizar el "corte" de la señal, se utilizó el programa Matlab, el cual nos permite hacer cortes de la señal de voz con mayor precisión que el programa empleado para su grabación. Dicho corte se llevó a cabo graficando la señal de voz y determinando visualmente el valor en el cual se superaba el ruido, a partir de este valor se guardaba una cantidad igual de muestras para todas las señales, desechando todos los demás datos.

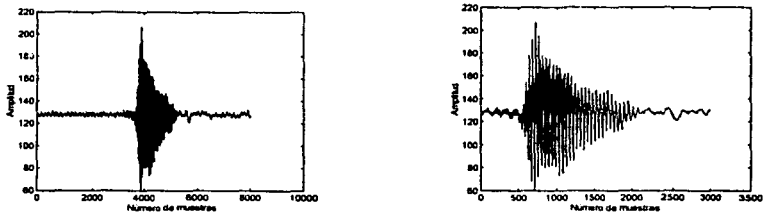


Figura 5.1. En las gráficas de arriba se muestra la señal de voz correspondiente a la palabra "fin" pronunciada por un hombre. La gráfica de la izquierda es la señal sin cortar y su número de muestras es de 7000, después del corte su tamaño se reduce a 3000 (gráfica de la derecha).

Para obtener los datos que serían introducidos a la red como patrones de prueba o de entrenamiento, las señales de voz fueron procesadas con alguno de los dos métodos descritos en el capitulo III.

Todos los procesamientos aplicados a las señales de voz se implementaron con funciones y rutinas del programa Matlab, se eligió este programa debido a que es fácil de utilizar, nos permite hacer modificaciones de forma rápida y además su sintaxis es parecida a algunos lenguajes de programación como el lenguaje C. Esta última característica es muy importante porque nos permitirá programar posteriormente, el método que nos proporcione la mejor respuesta de la red tomando como base las rutinas de Matlab sin realizar modificaciones significativas.

4.1 Entrenamiento de la red.

Existen varios programas para entrenar redes neuronales artificiales que tiene implementado el algoritmo de retropropagación del error, dentro de estos se encuentran Matlab y Dynamind. Para entrenar una red con estos programas se deben indicar los parámetros con los cuales se desea entrenarla, tales como el factor de aprendizaje, el error, el número de neuronas de la capa oculta, etc.

Matlab tiene varias funciones para el entrenamiento de redes, en todas ellas los parámetros de la red se pasan como argumentos de dichas funciones, mientras que en Dynamind los parámetros se deben proporcionar en tiempo de ejecución a través de menús.

Ambos programas reciben los patrones de la capa de entrada y de salida como un arreglo matricial, pero el formato que utilizan es diferente. En Matlab los datos de entrenamiento (patrones de entrenamiento así como sus respectivas salidas) se ponen en dos matrices, una con los patrones de entrenamiento y la otra con las salidas deseadas de la red. Estas matrices se guardan en archivos de texto.

La fase de entrenamiento termina cuando el error entre la salida que proporciona la red y la salida deseada es menor a un valor especificado. Al finalizar esta fase, los programas de entrenamiento guardan los pesos de las conexiones entre neuronas.

Ya sea que los pesos se obtuvieran con Dynamind o con Matlab, todos eran separados en cuatro archivos, dos con los pesos de las conexiones de las capas de salida y oculta y en los otros dos se guardaban los pesos de las neuronas de bias. Se hacia esta división porque todos los pesos eran probados con las funciones de Matlab.

Para determinar el número de neuronas de la capa oculta se empleó la siguiente formula:

$$NCO = \sqrt{(NCE)(NCS)} \qquad 5.1$$

donde :

NCO: Número de neuronas de la capa oculta.
NCE: Número de neuronas de la capa de entrada.
NCS: Número de neuronas de la capa de salida.

El valor de *NCO* se redondea al entero más pequeño y es solo un punto de partida con el cual se pueden realizar las primeras pruebas, porque como se dijo anteriormente no existe una regla bien definida para determinar el número de neuronas de la capa oculta que nos den el mejor resultado.

Con el propósito de tener un valor que determine el desempeño de la red y nos permita hacer comparaciones entre los resultados obtenidos con redes diferentes, se utilizó como parámetro la eficiencia, en este caso, definimos la eficiencia como el cociente del número de patrones que reconoció la red entre el número total de patrones. Es decir:

$$Eficiencia = \frac{Número\ de\ Patrones\ Reconocidos}{Número\ de\ Patrones\ Totales}$$

4.2 Pruebas con Vocales.

Las señales de voz más sencillas son las letras y para determinar si nuestro sistema sería capaz de reconocer al menos letras se realizaron pruebas con vocales, una vez que se lograra reconocer todas las vocales, se harían pruebas con palabras aisladas.

En estas pruebas se grabaron las vocales sin restricciones en su pronunciación, es decir no se le pidió al locutor que pronunciara rápido o despacio, con la finalidad de simular las condiciones reales en las cuales un locutor las emitiría al momento de utilizar el sistema.

De las ocho voces grabadas se eligieron solo seis, tres de ellas se emplearon para el entrenamiento y las otras tres para la fase de prueba. Una vez definidos ambos patrones se les aplicó un procesamiento para obtener los datos que fueron utilizados como entrada de la red. Para el caso de las vocales se utilizaron como procesamientos el espectro de la señal o los coeficientes LPC.

- 4.2.1 Entrenando la red con el espectro de la señal.

Las señales en el tiempo no nos proporcionan la información que la red necesita para una adecuada clasificación de palabras, además aún cuando la frecuencia de muestreo es relativamente pequeña (8 kHz), son muchas las muestras que se obtienen (el número de muestras que se tomaron para las vocales fue de 2000), por lo que se debe procesar la señal

para reducir el número de datos y también para obtener de ellas solo las características que sirvan a la red para que esta haga un correcto reconocimiento de todas las vocales.

Con el objeto de determinar los valores de eficiencia que se pueden obtener usando como entrada a la red el espectro en frecuencia, se aplicó a las señales la transformada de Fourier, la cual nos permite conocer el comportamiento de la señal con respecto a la frecuencia.

Se trabajó solo con la parte positiva del espectro, con ello se redujo a la mitad el número de patrones de entrada de la red. Como se dijo en el capitulo III, la mayor cantidad de información contenida en el espectro en frecuencia se localiza en los primeros mil hertz, razón por la cual se decidió entrenar a la red con los valores del espectro dentro del rango de 0-1000 Hz, teniendo con ello 375 datos de entrada por cada letra.

Por simplicidad se propuso como salida de la red una numeración binaria del 0 al 4, solo que en lugar de tomarse valores de 0 y 1, se tomaron entre -1 y 1 para facilitar el mapeo de la función de activación tangente hiperbólica, la cual tiene un rango de valores de –1 a 1.

Número	Correspondencia		
0	-1	-1	-1
1	-1	-1	1
2	-1	1	-1
3	-1	1	1
4	1	-1	-1

La topología de la red utilizada en este caso fue un perceptrón de dos capas como la mostrada en la figura 5.2.

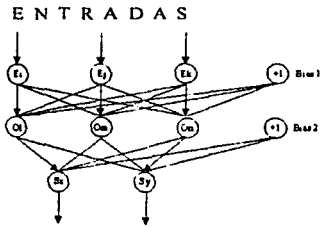


Figura 5.2 Perceptrón multicapa con dos capas.

Esta red se entrenó con el algoritmo de retropropagación del error. El número de neuronas de la capa intermedia se tomó como 33, este valor se obtuvo con la fórmula 5.1. Los parámetros iniciales de la red fueron los siguientes:

Número de Neuronas de la Capa de Entrada:	375
Número de Neuronas de la Capa Oculta:	33
Número de Neuronas de la Capa de Salida:	3
Factor de aprendizaje:	0.01
Error:	0.01

Los pares de patrones entrada-salida del conjunto de entrenamiento para cada persona fueron los siguientes:

Patrones de entrada	Patrones de salida		
Letra A	-1	-1	-1
Letra E	-1	-1	1
Letra I	-1	1	-1
Letra O	-1	1	1
Letra U	1	-1	-1

Dado que se utilizaron en el entrenamiento las voces de tres personas, la matriz de entrada de la red fue de 375 x 15 (tres personas x cinco vocales).

Una vez que se determinó la topología, el formato de los datos de entrada y los parámetros de la red, esta fue entrenada con la función *trainlm* (train feed-forward network with Levenberg-Marquardt) del programa Matlab utilizando como activación una función tangente hiperbólica.

Las pruebas consistieron en cambiar los parámetros de la red, es decir se cambió el número de neuronas de la capa oculta, el error o el factor de aprendizaje. Solo se hizo una modificación a la vez, para observar su influencia en la eficiencia de la red.

Después de varias pruebas el resultado obtenido no fue satisfactorio, es decir la red no logró reconocer todas las letras de los patrones de entrenamiento, reconociéndose en el mejor de los casos tres patrones de un total de quince.

No se realizaron más pruebas cambiando los parámetros, lo que se hizo fue graficar las señales de todas las vocales para observar sus diferencias, con ello se percibió que todas las señales de voz tienen una amplitud diferente, esto se debe a que el tono de voz cambia de una persona a otra y además a que algunas personas se acercan más al micrófono que otras.

Para disminuir estas diferencias, primero se eliminó la componente de corriente directa de la señal digital, restando esta componente a cada uno de sus valores de amplitud y dejando a la señal en una referencia de cero.

Una vez eliminado el valor constante de la señal, sus valores de amplitud fueron normalizados a valores entre –1 y 1 con el fin de obtener entradas que correspondieran con los valores de la función de activación (tangente hiperbólica en este caso).

Con estas modificaciones se logró un incremento significativo en el número de patrones reconocidos, no obstante no fue suficiente para conseguir un reconocimiento total de dichos patrones.

El reconocimiento de vocales es más simple que el reconocimiento de palabras y como el espectro en frecuencia no nos proporciona por si solo la información que la red necesita para hacer una adecuada clasificación de las vocales, ni siquiera en su fase de entrenamiento, entonces tampoco nos serviría para clasificar palabras, por esta razón este tipo de datos fueron descartados como patrones de entrada a la red y en su lugar se hicieron pruebas con el método de los coeficientes de predicción lineal.

- 4.2.2 Entrenando la red con los coeficientes de predicción lineal (LPC).

En estas pruebas se utilizó el método de los coeficientes de predicción lineal, el cual además de proporcionar una buena representación de la señal de voz nos permite reducir su tamaño en aproximadamente un 10%.

El método de los coeficientes de predicción se explicó con detalle en el capitulo anterior y por lo tanto no se repetirá el procedimiento para obtenerlos, sin embargo si es importante citar el tamaño de la ventana que se utilizó, así como el número de muestras de la señal antes de aplicar el método, pues estos dos valores influyen en el número total de coeficientes que serán introducidos a la red.

En este caso se utilizó una ventana de 90 ms, con lo cual obtuvimos un número de muestras por ventana igual a 720 (a 8kHz y 2000 muestras). Con estos valores y tomando doce coeficientes LPC por ventana, el número de datos de entrada fue de 60 elementos.

Una vez calculados los datos de entrada a la red, se tomaron los coeficientes correspondientes a la voz de seis personas, tres hombres y tres mujeres. De ellos se utilizaron los coeficientes que representaban la voz de dos hombres y una mujer para la fase de entrenamiento y como patrones de prueba se eligieron los tres restantes.

Al igual que las pruebas hechas con el espectro, la salida deseada de la red se tomó como una numeración binaria y fue entrenada nuevamente con la función *trainlm* de Matlab y los siguientes parámetros iniciales:

Número de neuronas de la capa de entrada:	60
Número de neuronas de la capa oculta:	13
Número de neuronas de la capa de salida:	3

Con los coeficientes de predicción lineal se lograron reconocer todos los patrones de entrenamiento, este resultado fue mucho mejor que el obtenido con el espectro en frecuencia si tomamos en cuenta que se logró con un número menor de datos de entrada de la red (con el espectro fueron necesarios 375 datos). Sin embargo, al probar los valores de los pesos de conexión con los patrones del conjunto de prueba, no se pudo obtener una eficiencia mayor a 0.6, como se muestra en la tabla 1, en donde el valor del factor de aprendizaje fue constante e igual a 0.01.

NCO	25	20	14	10	10	10	10
Error	0.01	0.01	1×10^{-4}	0.01	0.001	1×10^{-4}	0.001
Eficiencia	0.40	0.46	0.53	0.46	0.46	0.46	0.60

Tabla 1. En esta tabla se muestran los mejores resultados obtenidos en las pruebas con coeficientes LPC y el programa Matlab, en donde se aprecia que la máxima eficiencia obtenida fue de 0.6. (se reconocieron nueve patrones de prueba de un total de quince) NCO es el número de neuronas en la capa oculta.

Antes de hacer pruebas con los coeficientes ceptrum en vez de los coeficientes LPC o de cambiar más los parámetros de la red, se utilizó como programa de entrenamiento a Dynamind.

Para comparar los resultados obtenidos con Dynamind con aquellos que nos proporcionó Matlab, se realizaron las mismas pruebas, partiendo de los parámetros mostrados en la tabla 1.

Después de algunos entrenamientos utilizando Dynamind se logró una eficiencia unitaria, es decir la red fue capaz de reconocer todos los patrones de entrenamiento y de prueba. Lo anterior se consiguió con los siguientes parámetros:

Número de neuronas de la capa oculta:	20
Factor de aprendizaje:	0.1
Error:	0.01

4.3 Pruebas con palabras.

Una vez alcanzado el objetivo de reconocer las vocales, tanto del conjunto de prueba como del conjunto de entrenamiento, el siguiente paso consistió en determinar si esta arquitectura y programa de entrenamiento, en combinación con el método de coeficientes de predicción lineal podría ser generalizada y aplicada al caso de palabras aisladas. Para tal efecto, utilizamos cinco palabras: *casa, cosa, asa, perro y menos*, con

ellas se hicieron dos pruebas, una con palabras fonéticamente parecidas y otra con palabras fonéticamente diferentes.

Nuevamente se grabaron las voces de seis personas. dos mujeres y cuatro hombres, a estas señales se les aplicó el método de los coeficientes de predicción lineal y se tomaron las voces de dos hombres y una mujeres para la fase de entrenamiento y los tres restantes para la fase de prueba. El tamaño de la ventana, así como el número de muestras por señal se mantuvo constante.

1. Pruebas con palabras fonéticamente diferentes.

   Con el fin de determinar si los coeficientes de predicción nos servirían para reconocer palabras diferentes, se entrenó a la red con los parámetros con los cuales se lograron reconocer todas las vocales, es decir: número de neuronas de la capa oculta igual a 20, factor de aprendizaje igual a 0.1 y error igual a 0.01.

   Se Tomaron como datos de entrada los coeficientes de las palabras cosa, perro y menos.

   Los resultados a estas pruebas fueron satisfactorios, es decir se reconocieron todos los patrones del conjunto de entrenamiento y de prueba sin ningún problema y sin necesidad de modificar ningún parámetro.

2. Pruebas con palabras fonéticamente iguales.

   Aún cuando en la prueba anterior la red no tuvo ningún problema en reconocer palabras diferentes, era necesario comprobar si la red no confundiría palabras que se parecieran fonéticamente.

   Sin hacer ningún cambio en los parámetros de la red, esta se entrenó con las palabras: casa, cosa y asa, dando como resultado una correcta clasificación de todos los patrones del conjunto de prueba.

Con las pruebas anteriores, pudimos determinar que el perceptrón con una capa oculta era capaz de reconocer al menos tres palabras sin ningún problema. El siguiente paso fue comprobar si al incrementar el número de palabras se reconocerían nuevamente todos los patrones de prueba, para ello se grabaron veinte palabras; algunas de ellas se tomaron de las opciones de un procesador de texto, otras del menú de manipulación de archivos que se encuentra en cualquier programa y las demás fueron preposiciones de lugar. Estas palabras se eligieron solo para determinar si la red sería capaz de reconocer más de tres comandos y no definen una aplicación en particular.

Se pidió a ocho personas que pronunciaran las veinte palabras en las mismas condiciones en las que fueron grabadas las vocales. A estas señales de voz se les aplicó el método de los coeficientes de predicción lineal.

Como salida de la red se eligió nuevamente una numeración binaria de cero a diecinueve siguiendo la misma convención (rango de valores de -1 a 1). Dado que el número más grande de esta numeración es el: 1 -1 -1 1 1, se necesitaron cinco neuronas en la capa de salida para abarcar todo el rango de valores.

En esta ocasión el conjunto de pares entrada-salida para el conjunto de entrenamiento fue el siguiente:

Patrones de entrada	Patrones de salida				
cierra	-1	-1	-1	-1	-1
derecha	-1	-1	-1	-1	1
izquierda	-1	-1	-1	1	-1
sube	-1	-1	-1	1	1
baja	-1	-1	1	-1	-1
copia	-1	-1	1	-1	1
pega	-1	-1	1	1	-1
corta	-1	-1	1	1	1
espacio	-1	1	-1	-1	-1
fin	-1	1	-1	-1	1
inicio	-1	1	-1	1	-1
guarda	-1	1	-1	1	1
abre	-1	1	1	-1	-1
nuevo	-1	1	1	-1	1
mínimo	-1	1	1	1	-1
máximo	1	1	1	1	1
foco	1	-1	-1	-1	-1
itálica	1	-1	-1	-1	1
negrita	1	-1	-1	1	-1
raya	1	-1	-1	1	1

Lista 1

En el entrenamiento se utilizaron las voces de tres hombres y dos mujeres; las tres restantes se dejaron como patrones de prueba.

Cortamos las señales de voz tomando las primeras tres mil muestras, el tamaño de ventana utilizando fue igual a 90 ms (el mismo que se utilizó para vocales) por lo cual el número total de ventanas traslapadas fue igual a nueve. Dado que se toman los primeros doce coeficientes de predicción por cada ventana, el número de neuronas necesarias en la capa de entrada fue igual a 108 (doce coeficientes x nueve ventanas).

Como se emplearon las voces de cinco personas para la fase de entrenamiento y por cada palabra se utilizaron 108 datos para representarla, el tamaño de la matriz de entrada de la red fue de 108 x 100 (cinco personas x veinte palabras) .

Las pruebas consistieron nuevamente en cambiar los parámetros de la red para determinar con cuales podíamos lograr una mayor eficiencia. Primero se modificaba el número de neuronas de la capa oculta, después el error, el factor de aprendizaje y por último se intercambiaban algunos patrones de prueba por patrones de entrenamiento. El programa de entrenamiento fue Dynamind.

En todas las pruebas realizadas, la red logró reconocer solo algunos patrones de entrenamiento y ningún patrón de prueba.

En lugar de cambiar la topología de la red, se decidió normalizar las señales antes de aplicarles el método, para que todas tuvieran un valor de amplitud pico a pico de 256, es decir valores entre −128 y 128 en lugar de los valores entre −1 y 1 con los cuales se había estado trabajando. Se eligió el valor de 256 porque las palabras fueron grabadas utilizando una representación de 8 bits. La referencia a un nivel de cero se mantuvo para eliminar la componente de CD. En la siguiente figura se muestra una señal de voz antes y después de los cambios.

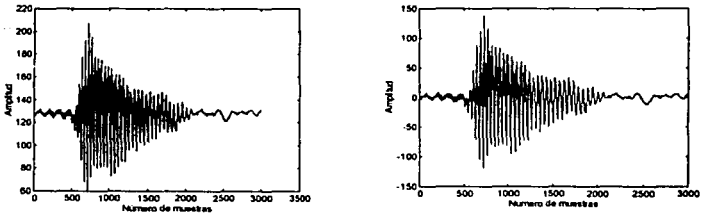
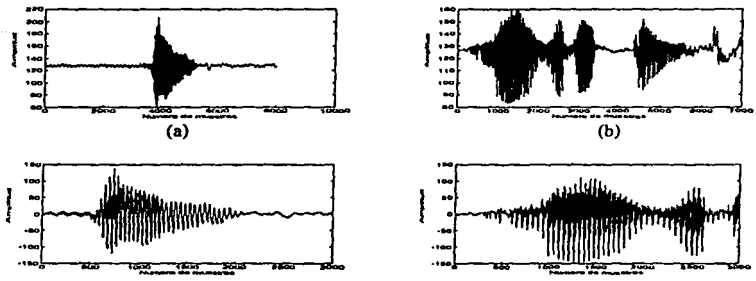


Figura 5.3. Señales de voz correspondientes a la palabra "fin" pronunciada por un hombre. Después de normalizarla y quitarle la componente de corriente directa obtenemos la señal de la derecha.

Después de normalizar las señales de voz, se les aplicó nuevamente el método de los coeficientes de predicción lineal. Con este cambio se logró que la red reconociera todos los patrones de entrenamiento, sin embargo al probar los pesos de conexión con los datos del conjunto de prueba, la red no pudo reconocer ninguno de dichos patrones.

Haciendo una revisión más minuciosa de las señales de voz, se observó que existen diferencias grandes entre las palabras de tres silabas y las de dos y una silaba en cuanto al número de muestras que tienen cada una. Palabras como "máximo" y "negrita", tienen una duración del doble o más de las palabras de una silaba, esta diferencia se refleja directamente en el número de muestras; por ejemplo la palabra "fin" tiene en promedio 2500 muestras, mientras que palabras de tres silabas como "negrita" tienen alrededor de 7500. Como se puede observar en la figura 5.4 la diferencia en el número de muestras es del triple.



(a) (b)

(c) (d)

Figura 5.4. Señales correspondientes a las palabras "fin" (a) y "negrita" (b) antes de ser cortadas. En la figura (d) se observa como se pierde información después de cortar la palabra "negrita" a 3000 muestras.

La diferencia en el número de muestras que existe entre las señales de tres silabas y las demás, es muy significativa y como todas las señales son cortadas a tres mil datos, se pierde más de la mitad de su información.

Tratando de evitar este problema se grabaron de nuevo las palabras de tres silabas, pero en esta ocasión se pidió a los locutores que las pronunciaran los más rápido posible. No obstante, esta modificación no fue suficiente porque las palabras de tres silabas seguían teniendo un mayor tamaño.

Había dos opciones: eliminar las palabras pequeñas y tomar un número mayor de muestras o eliminar las palabras grandes y seguir tomando los tres mil datos. Se optó por la segunda opción, ya que si tomamos un número mayor de muestras, el número de ventanas con traslape necesarias sería más grande y por consiguiente el número de datos de entrada de la red se incrementaría, haciendo los cálculos más lentos.

Se eliminaron todas las palabras de tres silabas y algunas palabras de dos silabas que tenían un numero grande de muestras. Se grabaron las palabras de dos y una silaba que se muestran en la siguiente lista:

cierra	copia	campo	abre
centro	tipo	dos	sube
baja	pega	corta	salva
fin	salir	sigue	menú
antes	casa	uno	todo

Lista 2

Cabe recordar que para estas palabras los patrones de salida son los mismos que se utilizaron con las palabras de la lista 1.

Se obtuvieron los coeficientes de predicción de todas las palabras de la lista anterior y se entrenó a la red, teniendo como resultado un reconocimiento total de los patrones de entrenamiento y algunos patrones del conjunto de prueba. En la tabla 2 se muestran los mejores resultados obtenidos en estas pruebas.

NCO	20	19	18	17
Error	0.01	0.001	0.10	0.001
Factor de aprendizaje	0.01	0.01	0.01	0.01
Eficiencia	0.13	0.18	0.20	0.15

Tabla 2. NCO es el número de neuronas de la capa oculta

Al eliminar las palabras de tres silabas la eficiencia de la red no fue nula, sin embargo fue muy baja.

Como se lograron reconocer todos los patrones de entrenamiento y algunos patrones de prueba, el siguiente paso consistió en realizar una serie de experimentos con la finalidad de aumentar la eficiencia. Las siguientes pruebas se realizaron tomando como datos de entrada a la red los valores obtenidos con el método de los coeficientes de predicción lineal o con el método de los coeficientes cepstrum en escala mel, para determinar con cual de ellos se obtenía la mayor eficiencia.

- 4.3.1 Pruebas con los coeficientes de predicción lineal.

a) Normalizando los coeficientes de predicción lineal.

Esta prueba se dividió en dos partes, en la primera se tomaron las señales de voz sin normalizar, se obtuvieron los coeficientes de predicción correspondientes y estos se mapearon a valores entre -1 y 1 (el mismo rango de valores de la función sigmoidal) para establecer que valores de eficiencia se obtenían. En la segunda prueba se volvieron a normalizar los coeficientes de predicción y además se normalizaron las señales de voz dentro del intervalo de −128 a 128. En la siguiente tabla se compara los valores de eficiencia obtenidos en esta prueba con las eficiencias de la tabla 2.

NCO y Error	20	19	18	17
	0.01	0.001	0.10	0.001
Factor de aprendizaje	0.01	0.01	0.01	0.01
Eficiencia (tabla 2)	0.13	0.18	0.20	0.15
Eficiencia con coeficientes normalizados	0.13	0.13	0.16	0.13
Eficiencia normalizando señal y coeficientes	0.20	0.20	0.28	0.21

Tabla 3. Como se puede observar, la eficiencia de la red aumentó ligeramente cuando se normalizaron tanto la señal como los coeficientes de predicción.

La eficiencia de la red aumentó ligeramente cuando se normalizaron tanto los coeficientes de predicción lineal como la señal de voz, si bien este aumento no fue grande si fue mayor a los obtenidos anteriormente.

Tomando en cuenta que se logra una mayor eficiencia de la red normalizando los valores de la señal de voz antes del aplicarle el método de los LPC y normalizando los coeficientes, se decidió hacer lo mismo para las siguientes pruebas.

b) Disminuyendo el tamaño de la ventana.

Se entrenó a la red manteniendo constantes sus parámetros (número de neuronas de la capa oculta dado por la formula 5.1, un factor de aprendizaje igual a 0.1 y un error igual a 0.01) y se modificó solo el tamaño de la ventana en el proceso de segmentación, para observar como cambia la eficiencia de la red al alterar dicho valor.

El tamaño de la ventana se disminuyó en 5ms, partiendo de 85ms y llegando a un valor de 60 ms. No se pudieron hacer pruebas con un tamaño de ventana menor, porque se tuvieron problemas con la memoria virtual de la computadora cuando se utilizaron tamaños de matriz de entrada mayores a 156 x 100 (obtenido con un tamaño de ventana de 60 ms).

La tabla 4 contiene los resultados de esta prueba.

PRUEBA	Eficiencia
tamaño de la ventana igual a 85 ms.	0.16
tamaño de la ventana igual a 75 ms.	0.18
tamaño de la ventana igual a 65 ms.	0.20
tamaño de la ventana igual a 60 ms.	0.20

Tabla 4

Como se puede observar, la eficiencia aumentó ligeramente cuando disminuimos el tamaño de la ventana.

c) Entrenando a la red con dos capas intermedias.

Modificamos la topología de la red neuronal empleada hasta este momento. En lugar de utilizar una capa intermedia se utilizaron dos, tal como se muestra en la figura 5.5.

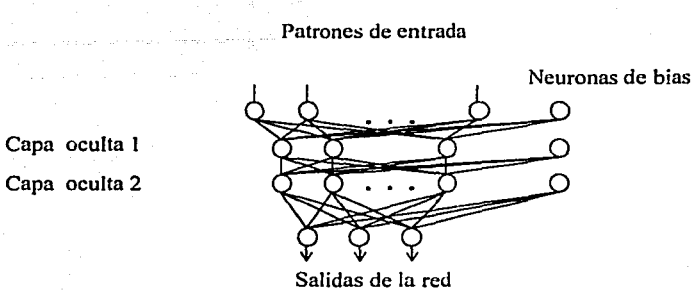


Figura 5.5 Red neuronal con dos capas ocultas

En la prueba se dejó constante el factor de aprendizaje y se modificaron el error y el número de neuronas de una de las capas ocultas, dejando constante el número de neuronas de la otra capa.

NCO 1	20	20	20	20	20	20	20	27	17	14
NCO 2	20	25	15	30	10	25	25	20	20	20
Error	0.001	0.001	0.01	0.01	0.01	0.1	0.01	0.1	0.1	0.1
Eficiencia	0.0	0.18	0.18	0.20	0.15	0.00	0.15	0.0	0.0	0.0

Tabla 5. NCO1 es el número de neuronas de la primer capa oculta. NCO2 es el numero de neuronas de las segunda capa oculta.

La eficiencia de la red no mejoró al agregar una capa oculta; como se puede observar en la tabla 5, este valor no es mayor de 0.2.

Para veinte palabras la eficiencia difícilmente fue mayor a 0.3. Este valor no se logró superar aún cuando se modificaron los parámetros de la red y se utilizaron dos capas ocultas.

Dado que la red fue capaz de reconocer sin problema tres comandos se decidió disminuir el número de palabras que se debía reconocer, para observar los valores de eficiencia que se obtienen.

d) Reduciendo el número de comandos.

Se eligieron los coeficientes correspondientes a doce palabras de la lista 2 y se dividieron en dos conjuntos de seis comandos cada uno, con cada uno de estos conjuntos se entrenó a red para determinar cual de ellos reconocía mejor la red.

Las palabras del primer conjunto fueron: *tipo, menú, salva, sigue, baja y casa*, mientras que las del segundo fueron: *sube, fin, todo, dos, abre y campo*. Además de probar estos dos conjuntos, se intercambiaron palabras de un conjunto y otro.

Después de varias pruebas, se encontró que las palabras que mejor se reconocieron fueron las del segundo conjunto (*sube, fin, todo, dos, abre y campo*), lográndose una eficiencia máxima de 0.41. En la siguiente tabla se muestran los mejores resultados obtenidos en esta prueba.

NCO	31	28	25	22
Error	0.01	0.01	0.01	0.01
Eficiencia	0.33	0.25	0.41	0.41

Tabla 6

Con el objeto de hacer una comparación minuciosa de las seis señales de voz de los conjuntos de prueba y entrenamiento, estas se graficaron, notándose que no solo es diferente el intervalo de tiempo en el que dos personas pronuncian una misma palabra, sino también que existen variaciones en el intervalo que hay entre las sílabas. Estas variaciones pueden tener influencia en el entrenamiento de la red y provocar que esta no sea capaz de reconocer de forma eficaz los datos de prueba. Este problema se ilustra en la figura 5.6

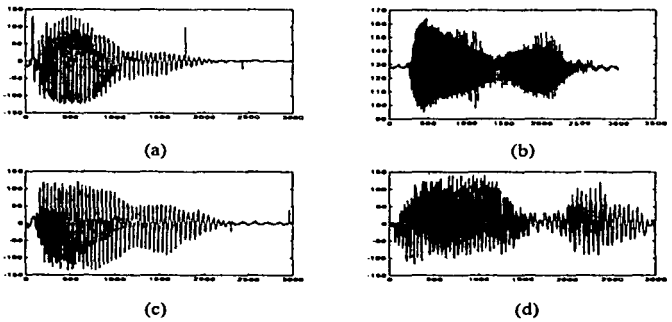


Figura 5.6 Las señales mostradas en esta figura corresponden a la palabra "pega". Las señales (a) y (b) fueron pronunciadas por dos personas diferentes, como se observa la persona que pronunció la palabra en (b) se tardó más en pronunciar la segunda sílaba. En la gráfica (d) ocurre un fenómeno similar, solo que en esta ocasión ambas palabras (c) y (d) fueron pronunciadas por la misma persona.

Con el fin de determinar que tanto influye este fenómeno en la eficiencia de la red, se decidió que todas las palabras fueran de una sílaba. Se eligieron palabras de una sílaba porque estas tienen variaciones menores respecto a las palabras de dos.

Se grabaron ocho palabras: *fin, dos, ven, con, tres, por, si y no*, las cuales fueron pronunciadas por ocho locutores, cinco hombres y tres mujeres. Para los patrones de entrenamiento se tomaron las voces de cinco personas y las tres restantes se utilizaron para probar la eficiencia de la red. Además se redujo a tres el número de neuronas de salidas.

En este caso los pares entrada-salida del conjunto de entrenamiento fueron los siguientes:

Patrones de entrada	Patrones de salida		
fin	-1	-1	-1
dos	-1	-1	1
ven	-1	1	-1
con	-1	1	1
tres	1	-1	-1
por	1	-1	1
si	1	1	-1
no	1	1	1

Lista 3

e) Disminuyendo el tamaño de la ventana.

Esta prueba ya se había realizado, sin embargo debido a problemas con la memoria virtual el tamaño de ventana más pequeño para veinte palabras fue de 60ms. En esta ocasión se realizaron pruebas con dos tamaños de ventana (64 y 32 ms) para comparar la eficiencia que se obtiene con uno y otro.

Tomando como tamaño de ventana el valor de 64 ms y dejando constantes el error (0.01) y el factor de aprendizaje (0.1) obtuvimos:

NCO	3	4	5	6	7	8	9	10	11	12	13	14	15
Eficiencia	0.33	0.35	0.35	0.33	0.35	0.35	0.45	0.43	0.40	0.33	0.33	0.40	0.45

Tabla 7

Reduciendo ahora el tamaño de la ventana a 32 ms:

NCO	16	17	18	19	20	21	22	23	24	25	26	27	28
Eficiencia	0.40	0.35	0.35	0.33	0.43	0.30	0.33	0.30	0.45	0.40	0.43	0.30	0.43

Tabla 8

En este caso se obtuvieron prácticamente los mismos valores de eficiencia sin importar el tamaño de la ventana. Pero se siguió empleando el valor de 32ms para las siguientes pruebas para obtener una mayor resolución en el espectro.

f) Distribuyendo los datos de entrada en varias redes.

Se cambió la arquitectura de la red neuronal o mejor dicho se utilizaron tres redes en lugar de una. Dos redes fueron usadas como entrada y sus salidas se tomaron como entrada de la tercera, tal como se muestra en la figura 5.7.

Dado que se usaron dos redes de entrada, los coeficientes de predicción que representan a la señal de voz también tuvieron que ser divididos; los primeros datos fueron tomados como entrada de una de las redes y la otra mitad como entrada de la segunda red, en ambos casos los datos de salida deseados fueron los mismo.

Cada una de estas redes fue entrenada por separado, las salidas generadas por las dos redes de entrada se agruparon en un solo vector y este a su vez fue tomado como entrada de la tercera red (red de salida). Para las tres redes los valores de salida deseados fueron los mismos de la lista 3, sin embargo a los patrones de entrenamiento de la red de salida se le agregaron variaciones.

Por ejemplo para las palabras "fin" y "no", los patrones de entrenamiento para la tercera red fueron:

	Patrón de entrada						Patrón de salida deseado		
	primera mitad de los coeficientes			segunda mitad de los coeficientes					
	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0
fin	-0.9	-0.9	-0.9	-0.9	-0.9	-0.9	-0.9	-0.9	-0.9
	-0.8	-0.8	-0.8	-0.8	-0.8	-0.8	-1.0	-0.8	-0.8
	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
no	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
	0.8	0.8	0.8	0.8	0.8	0.8	1.0	0.8	0.8

Después de entrenar la red de salida, no se modificó ninguno de sus parámetros ni tampoco sus pesos. Los cambios fueron hechos solamente en las dos redes de entrada.

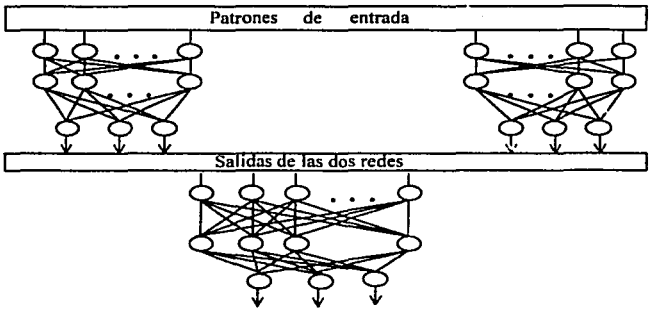


Figura 5.7. División de la red neuronal en dos redes independientes y una tercera cuya entrada depende de la salida de las dos anteriores. Cada red es entrenada por separado.

Tomando la arquitectura de red presentada en la figura anterior, se hicieron tres tipos de pruebas: en una de ellas se dejó constante en número de neuronas ocultas de la segunda red, en la siguiente se dejó constante el número de neuronas de la primera red y por último se modificaron las neuronas de ambas redes. En los tres casos se mantuvieron constantes el factor de aprendizaje (0.1) y el error (0.01).

Dejando en dieciocho el número de neuronas de la capa oculta de la segunda red (la que contiene los últimos datos del vector de coeficientes de predicción lineal) y manteniendo este valor constante, se hicieron cambios en el número de neuronas de la primera red, con lo cual se obtuvieron los siguientes resultados:

NCO	16	17	18	19	20	21	22	23	24	25	26	27	28
Eficiencia	0.35	0.33	0.33	0.35	0.40	0.35	0.35	0.45	0.35	0.35	0.33	0.35	0.33

Tabla 9. Eficiencia del conjunto de redes, dejando en 18 el número de neuronas de la capa oculta de la segunda red.

Ahora, dejando en dieciocho el número de neuronas de la capa oculta de la segunda red y variando el número de neuronas de la primera obtuvimos:

NCO	16	17	18	19	20	21	22	23	24	25	26	27	28
Eficiencia	0.40	0.35	0.35	0.33	0.43	0.30	0.33	0.30	0.45	0.40	0.43	0.30	0.43

Tabla 10. Eficiencia del conjunto de redes, dejando en 18 el número de neuronas de la capa oculta de la primera red.

Por último, se modificaron las capas ocultas de las dos redes, pero poniendo en ambas el mismo número de neuronas:

NCO	16	17	18	19	20	21	22	23	24	25	26	27	28
Eficiencia	0.35	0.35	0.33	0.40	0.33	0.35	0.35	0.40	0.33	0.30	0.30	0.33	0.30

Tabla 11. Eficiencia del conjunto de redes, modificando el número de neuronas de la capa oculta de las dos redes de entrada.

Como se puede apreciar en las tablas 9, 10 y 11 los valores de eficiencia que se obtienen al distribuir los datos de entrada en tres redes son prácticamente los mismos que se encuentran en la tabla 7, los cuales fueron obtenidos más fácilmente con una sola red.

Al utilizar como datos de entrada los coeficientes de predicción lineal de palabras de una sílaba, la máxima eficiencia fue de 0.45. Este valor se obtuvo con una red de una capa oculta de nueve neuronas, un error igual a 0.01, un factor de aprendizaje igual a 0.1 y empleando ventanas de 64 ms. Ahora, para determinar si esta eficiencia aumenta o disminuye al incrementar el número de palabras, se entrenó a la red con un número mayor de palabras, modificando el número de neuronas de la capa oculta y dejando constantes todos los demás parámetros. La máxima eficiencia obtenida en cada caso se muestra en la siguiente tabla:

NÚMERO DE COMANDOS	EFICIENCIA
10	0.38
11	0.38
12	0.28
13	0.28
14	0.20
15	0.26
16	0.19

Tabla 12

El caso en el cual el número de palabras se incrementó a 16 fue especial, pues se tuvo que aumentar a cinco el número de neuronas de la capa de salida.

Como se puede apreciar en la tabla 12, la eficiencia disminuye a medida que el número de palabras aumenta.

- 4.3.2 Pruebas con los coeficientes cepstrum en escala mel (MFCC).

Al igual que el método de los coeficientes de predicción lineal, el método para obtener los coeficientes MFCC se describió con detalle en el capítulo III, solo cabe recordar que este método se implementó con rutinas para el programa Matlab. Sin embargo el entrenamiento se realizó con el programa Dynamind.

Para hacer una comparación entre los valores de eficiencia que se obtiene con el método de los coeficientes cepstrum en escala mel y los coeficiente de predicción lineal, se repitieron las mismas pruebas. Las salidas que se esperaba obtener de la red también fueron las mismas.

a) Normalizando los coeficientes cepstrum.

Se utilizaron las palabras de la lista 2 y se dividió la prueba en dos partes, en la primera se tomaron las señales de voz sin normalizar, se obtuvieron los coeficientes cepstrum y estos se mapearon a valores entre -1 y 1 para determinar que valores de eficiencia se obtenían. En la segunda prueba se normalizaron nuevamente los coeficientes y además se normalizaron las señales de voz a valores entre –128 y 128. El tamaño de la ventana en este caso fue de 90 ms.

NCO y Error	20 0.01	19 0.001	18 0.10	17 0.001
Factor de aprendizaje	0.01	0.01	0.01	0.01
Eficiencia	0.08	0.12	0.13	0.09
Eficiencia con coeficientes Normalizados	0.07	0.08	0.06	0.08
Eficiencia normalizando señal y coeficientes	0.18	0.16	0.12	0.14

Tabla 13

Al igual que los coeficientes de predicción lineal se puede observar en la tabla anterior, un ligero aumento en la eficiencia debido a la normalización tanto de las señales como de los coeficientes MFCC.

b) Disminuyendo el tamaño de la ventana.

En esta prueba se entrenó a la red manteniendo constante el número de neuronas de la capa oculta, el factor de aprendizaje (0.1), el error (0.01) y se modificó el tamaño de la ventana de Hamming utilizada en el ventaneo con traslape.

El tamaño de la ventana se disminuyó de 85ms hasta 60 ms. No se pudieron tomar valores más pequeños porque, al igual que los coeficientes de predicción lineal se tuvieron problemas con la memoria virtual de la computadora para esta cantidad de palabras.

PRUEBA	Eficiencia
tamaño de la ventana igual a 85 ms.	0.05
tamaño de la ventana igual a 75 ms.	0.02
tamaño de la ventana igual a 65 ms.	0.02
tamaño de la ventana igual a 60 ms.	0.02

Tabla 14

A diferencia de los valores obtenidos con los coeficientes de predicción lineal. la eficiencia se mantuvo prácticamente constante.

c) Entrenando a la red con dos capas intermedias.

Se cambió un poco la topología de la red neuronal agregándole una capa oculta, como se muestra en la figura 5.5. Mantuvimos constante el factor de aprendizaje y se modificaron el error y el número de neuronas de una de las capas ocultas, dejando constante el número de neuronas de la otra capa.

NCO 1	20	20	20	20	20	20	20	27	17	14
NCO 2	20	25	15	30	10	25	25	20	20	20
Error	0.001	0.001	0.01	0.01	0.01	0.1	0.01	0.1	0.1	0.1
Eficiencia	0.02	0.02	0.00	0.00	0.00	0.00	0.00	0.0	0.0	0.0

Tabla 15. Eficiencia de la red de dos capas. NCO1 es el número de neuronas de la primer capa oculta, NCO2 es el número de neuronas de las segunda capa oculta.

En este caso la eficiencia de la red disminuyó a tal grado que no se pudieron reconocer todos los patrones de entrenamiento.

d) Reduciendo el número de comandos.

En esta prueba se redujo el número de palabras, pero en este caso no se tomaron doce, sino solo las seis palabras que mejor se reconocieron en las pruebas con coeficientes de predicción lineal (*sube, fin, todo, dos, abre y campo).*

NCO	31	28	25	22
Error	0.01	0.01	0.01	0.01
Eficiencia	0.25	0.15	0.50	0.15

Tabla 16

Con se puede apreciar en la tabla anterior, se logró obtener una eficiencia de 0.5, este valor es mayor que cualquiera que nos haya proporcionado los coeficientes LPC.

Tomando en cuenta el problema mostrado en la figura 5.6, se redujo el número de comandos a ocho palabras de una silaba (*fin, dos, ven,con, tres, por, si y no*), también se redujo a tres el número de neuronas de la capa de salidas.

e) Disminuyendo el tamaño de la ventana.

Se realizaron pruebas con dos tamaños de ventana (64 y 32 ms) para comparar la eficiencia que se obtiene con uno y otro.

Tomando como tamaño de ventana el valor de 64ms y dejando constantes el error (0.01) y el factor de aprendizaje (0.1) los valores de eficiencia fueron:

NCO	3	4	5	6	7	8	9	10	11	12	13	14	15
Eficiencia	0.50	0.38	0.50	0.50	0.50	0.44	0.50	0.56	0.44	0.50	0.56	0.50	0.55

Tabla 17

Ahora, disminuyendo el tamaño de la ventana a la mitad:

NCO	16	17	18	19	20	21	22	23	24	25	26	27	28
Eficiencia	0.19	0.44	0.56	0.69	0.75	0.63	0.56	0.44	0.63	0.44	0.38	0.63	0.44

Tabla 18

En la tabla 14 se observó que la eficiencia se mantuvo prácticamente constante sin importar el tamaño de la ventana, sin embrago esta mejoró mucho cuando se utilizaron los coeficientes ceptrum en escala mel con un tamaño de ventana igual a 32ms y se redujo el número de palabras. Lográndose obtener incluso un valor de 0.75, el máximo obtenido hasta ahora con ocho comandos.

f) Distribuyendo los datos de entrada en varias redes.

Ahora se cambió la arquitectura de la red neuronal, dividiéndola en tres redes como se muestra en la figura 5.7.

Se dividió cada vector de coeficientes en dos y cada mitad se utilizó como entrada de una de las dos redes de entrada. Todas la redes fueron entrenadas por separado.

Dejando constante el número de neuronas de la capa oculta de la segunda red, el factor de aprendizaje (0.1) y el error (0.01), se cambió el número de neuronas de la primera red:

NCO	16	17	18	19	20	21	22	23	24	25	26	27	28
Eficiencia	0.44	0.56	0.62	0.69	0.37	0.50	0.50	0.56	0.44	0.50	0.44	0.44	0.44

Tabla 19. Eficiencia del conjunto de redes, tomando como número de neuronas de la capa oculta de la segunda red igual a 18.

Dejando ahora constante el número de neuronas de la capa oculta de la segunda red y variando el número de neuronas de la primera:

NCO	16	17	18	19	20	21	22	23	24	25	26	27	28
Eficiencia	0.69	0.69	0.69	0.69	0.56	0.50	0.62	0.50	0.62	0.68	0.56	0.62	0.62

Tabla 20. Eficiencia del conjunto de redes, tomando como número de neuronas de la capa oculta de la primera red igual a 18.

Modificando el número de neuronas de la capa oculta de ambas redes:

NCO	16	17	18	19	20	21	22	23	24	25	26	27	28
Eficiencia	0.50	0.44	0.38	0.44	0.50	0.56	0.50	0.37	0.44	0.31	0.31	0.62	0.56

Tabla 21. Eficiencia del conjunto de redes, tomando igual número de neuronas de la capa oculta para la primera y segunda red.

Aun cuando los valores de eficiencia son relativamente altos, en ninguna ocasión se obtuvo una eficiencia de 0.75, la cual se logró más fácilmente con una sola red.

Para determinar si la eficiencia de la red aumenta o disminuye al incrementar el número de palabras, se hicieron varias pruebas, manteniendo constante el tamaño de ventana (igual a 32 ms), el error (0.01), el factor de aprendizaje (0.01) y modificando el número de neuronas de la capa oculta. La máxima eficiencia obtenida en cada caso se muestra en la tabla 22.

NÚMERO DE COMANDOS	EFICIENCIA
10	0.58
11	0.58
12	0.38
13	0.34
14	0.32
15	0.36
16	0.28

Tabla 22

Un caso especial fue aquel en el cual la red se entrenó con 16 palabras, pues se tuvo que incrementar a cinco el número de neuronas de la capa de salida.

Al igual que los LPC, con los coeficientes cepstrum la eficiencia disminuyó a medida que se aumentó el número de comandos.

La máxima eficiencia que se pudo obtener con los coeficientes cepstrum fue 0.75 y se logró con los siguientes parámetros: Número de neuronas de la capa oculta igual a veinte, factor de aprendizaje igual a 0.1, error igual a 0.01 y un tamaño de ventana igual a 32 ms.

- 4.3.3 Comparación de las eficiencias y tiempos de entrenamiento entre Dynamind y Matlab.

Con el objeto de hacer una comparación del tiempo de entrenamiento y valores de eficiencia que se obtienen cuando la red se entrena con el programa Dynamind o las funciones de Matlab, se realizaron pruebas con diferentes valores de error y factor de aprendizaje. Además se entrenó a la red con dos tipos de funciones sigmoidales: tangente hiperbólica y logarítmica.

En cada caso la red se entrenó con los coeficientes de ocho palabras y se mantuvo el número de neuronas de la capa oculta igual a veinte.

Las funciones de Matlab que con las cuales se entrenó la red fueron las siguientes:

Trainlm: Train feed-forward network with Levenverg-Marquardt.
Trainbpx: Train feed-forward network with fast backpropagation.
Trainbpm: Train feed-forward network with fast backpropagation + momentum
Trainbpa: Train feed-forward network with fast backpropagation + adaptive learning.

La función con la cual se habían estado haciendo las pruebas fue la sigmoidal tangente hiperbólica, por lo cual no se modificaron los datos de entrada a la red, ni se cambiaron los datos de salida.

Manteniendo constantes el error (0.01) y el factor de aprendizaje (0.1):

Programa / función de entrenamiento	Dynamind	Matlab: **Trainlm**	Matlab: **Trainbpx**	Matlab: **Trainbpm**	Matlab: **Trainbpa**
Tiempo aprox. de entrenamiento	10 min	3.5 hrs	15 min	4.5 hrs	20 min
Eficiencia con LPC	0.45	0.38	0.40	0.10	0.30
Eficiencia con MFCC	0.75	0.75	0.56	0.00	0.59

Como se puede observar, la máxima eficiencia se obtuvo con los coeficientes cepstrum. Tanto Dynamind como la función Trainlm de Matlab nos permiten obtener una eficiencia de 0.75, sin embargo con el primero se obtiene un entrenamiento más rápido.

Cambiando la función de activación tangente hiperbólica por la función logarítmica:

Dado que la función sigmoidal logística mapea los valores dentro del rango de 0 a 1. se tuvieron que normalizar los valores de los coeficientes de cero a uno, en vez de -1 a 1, además modificamos los patrones de salida de la red dentro de este mismo rango.

Haciendo las modificaciones anteriores, se mantuvieron constantes el error (0.01) y el factor de aprendizaje (0.1), con lo cual obtuvimos:

Programa / función de entrenamiento	Dynamind	Matlab: **Trainlm**	Matlab: **Trainbpx**	Matlab: **Trainbpm**	Matlab: **Trainbpa**
Tiempo aprox. de entrenamiento	30 min	4 hrs	15 min	4 hrs	10 min
Eficiencia con LPC	0.20	0.33	0.40	0.00	0.30
Eficiencia con MFCC	0.40	0.56	0.60	0.00	0.59

Con la función logarítmica no se logró obtener una eficiencia mayor a 0.6.

En las siguientes pruebas se utilizó una función de activación tangente hiperbólica y se mantuvieron constantes todos los parámetros de la red a excepción del factor de aprendizaje, el cual se varió de 0.001 a 1:

Programa / función de entrenamiento	Dynamind	Matlab: **Trainlm**	Matlab: **Trainbpx**	Matlab: **Trainbpm**	Matlab: **Trainbpa**
FA = 0.01					
TAE	5 min	4.5 hrs	15 min	4 hrs	20 min
Eficiencia con LPC	0.48	0.20	0.20	0.00	0.40
Eficiencia con MFCC	0.63	0.40	0.62	0.00	0.56
FA = 0.001					
TAE	5 min	3.5 hrs	15 min	3 hrs	10 min
Eficiencia con LPC	0.30	0.00	0.48	0.00	0.35
Eficiencia con MFCC	0.56	0.00	0.50	0.00	0.56
FA = 1					
TAE	2 hrs	3.5 hrs	15 min	3 hrs	10 min
Eficiencia con LPC	0.00	0.00	0.34	0.00	0.40
Eficiencia con MFCC	0.00	0.00	0.56	0.00	0.69

TEA = tiempo aproximado de entrenamiento
FA = factor de aprendizaje

Se puede apreciar que al disminuir el factor de aprendizaje la eficiencia disminuye ligeramente, sin embargo cuando el factor de aprendizaje es unitario, el programa

Dynamind y la función trainlm no son capaces de reconocer ningún patrón, además de aumentar significativamente el tiempo de entrenamiento del programa Dynamind.

Modificando ahora el error de $1.X10^{-3}$ a $1.X10^{-4}$ y dejando constantes todos los demás parámetros de la red:

Programa / función de entrenamiento	Dynamind	Matlab: **Trainlm**	Matlab: **Trainbpx**	Matlab: **Trainbpm**	Matlab: **Trainbpa**
Error = $1.X10^{-3}$					
TAE	20 min	5 hrs	35 min	4.5 hrs	40 min
Eficiencia con LPC	0.50	0.00	0.33	0.00	0.48
Eficiencia con MFCC	0.75	0.00	0.65	0.00	0.62
Error = $1.X10^{-4}$					
TAE	45 min	5 hrs	40 min	4.5 hrs	40 min
Eficiencia con LPC	0.40	0.00	0.34	0.00	0.20
Eficiencia con MFCC	0.75	0.00	0.50	0.00	0.62

Se obtienen prácticamente los mismos valores de eficiencia sin importar el valor del error, pero cuando aumentamos el error, aumenta el tiempo de entrenamiento.

En la mayoría de las pruebas, el método de los coeficientes cepstrum en combinación con el programa de entrenamiento Dynamind nos proporcionaron el mejor tiempo de entrenamiento y la mayor eficiencia.

- Redes neuronales con retraso en el tiempo.

Como se dijo en el capitulo II, otra arquitectura empleada en los sistemas de reconocimiento de voz es la red neuronal con retrazo en el tiempo o TDNN, en este tipo de redes las unidades básicas (neuronas) incluyen en sus sinapsis un filtro. con el cual se toman en cuenta las características dinámicas de las señales de voz.

Las redes TDNN se entrenan con una extensión del algoritmo de retropropagación del error llamado retropropagación a través del tiempo. Este algoritmo no se implementó, sin embargo como se pretende que este texto sirva de referencia para futuros proyectos relacionados con los sistemas de reconocimiento de voz, cabe mencionar que existe un programa llamado NICO; el cual permite construir redes con retraso en el tiempo y entrenarlas con el algoritmo de retropropagación a través del tiempo.

NICO es un programa de distribución libre que se puede obtener a través de Internet en la dirección: http://www.speech.kth.se/NICO/index.html y corre en ambientes Unix.

CAPITULO V

DISCUSIÓN

DISCUSIÓN.

Al entrenar la red con vocales se observó que el espectro en frecuencia no proporciona la información necesaria para que la red pueda distinguir las diferentes señales de voz. Aún cuando se lograron reconocer todos los patrones de entrenamiento, fueron pocas las vocales del conjunto de prueba que se reconocieron (solo tres patrones de un total de quince).

Debido a estos resultados se optó por emplear los coeficientes de predicción lineal como datos de entrada de la red; este cambio, junto con el empleo del programa de entrenamiento Dynamind en lugar de Matlab nos permitió reconocer todos los patrones de entrenamiento y de prueba.

Una vez que se logró que la red reconociera todas las vocales, se decidió hacer pruebas para determinar si con esta topología la red podría reconocer no solo vocales sino también palabras. Para tal efecto se realizaron pruebas con tres palabras parecidas y tres diferentes, en ambos casos la eficiencia obtenida fue unitaria.

Dado que pudimos reconocer tres palabras sin ningún problema, se pensó que la red sería capaz de identificar un número mayor empleando la misma topología y el método de los coeficientes de predicción lineal. Sin embargo, al aumentar el número de palabras, pasando de tres a veinte, la red no fue capaz de reconocer siquiera la mitad del conjunto de entrenamiento.

Para mejorar este resultado, se realizaron varias pruebas en las cuales se cambiaron los datos de entrada de la red utilizando los coeficientes de predicción lineal o los coeficientes cepstrum, también se cambio la topología de la red, pasando de una red de una capa oculta a redes con dos capas ocultas o incluso redes divididas.

Debido a que la duración de una palabra de tres silabas es mayor que la de dos y una silaba, se decidió eliminar las palabras de tres silabas. Al eliminar estos comandos, se logró que la red hiciera una correcta clasificación de las veinte palabras del conjunto de entrenamiento, sin embargo la eficiencia de la red para los patrones de prueba fue muy baja, con se puede notar en la tabla 2 del capitulo anterior, la cual se reproduce a continuación:

NCO	20	19	18	17
Error	0.01	0.001	0.10	0.001
Factor de aprendizaje	0.01	0.01	0.01	0.01
Eficiencia	0.13	0.18	0.20	0.15

Además de normalizar las señales de voz se normalizaron los coeficientes, lográndose con ello un ligero aumento en la eficiencia de la red, como se puede observar en las tablas 5.3 y 5.13.

Es decir, la red tuvo un mejor desempeño cuando sus entradas se encontraban dentro del rango de valores (de –1 a 1) que puede tomar la función de activación sigmoidal tangente hiperbólica.

Cuando entrenamos a la red con veinte palabras, se disminuye la duración de la ventana y se emplean como datos de entrada los coeficientes de predicción lineal la eficiencia aumenta, sin embargo esto no fue así con los coeficientes cepstrum, con los cuales la eficiencia se mantuvo prácticamente constante.

Como no se logró reconocer en la fase de prueba a las veinte palabras propuestas, se decidió entrenar a la red con menos palabras. Como se puede apreciar en la gráfica de la figura 6.1, la eficiencia de la red fue mayor cuando se redujo a ocho el número de palabras.

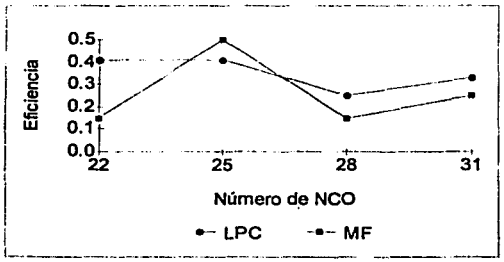


Figura 6.1. Eficiencia obtenida con ocho comandos de voz. Aun cuando la eficiencia en estas pruebas apenas superó el valor de 0.4 en general fue mayor que los valores obtenidos en las pruebas anteriores.

Como se dijo en el capitulo I, existen variaciones en la voz debidas a diversos factores tales como el cambio en el estado de animo de la persona o el acento de una región geográfica.

Se graficaron todas las palabras con las cuales se estaba entrenando la red y se observó que el intervalo de tiempo en el cual una persona pronuncia dos silabas varía no solo de una persona a otra, sino que además las palabras de dos silabas pronunciadas por una misma persona presenta cambios. Este problema se ilustra en la figura 6.2.

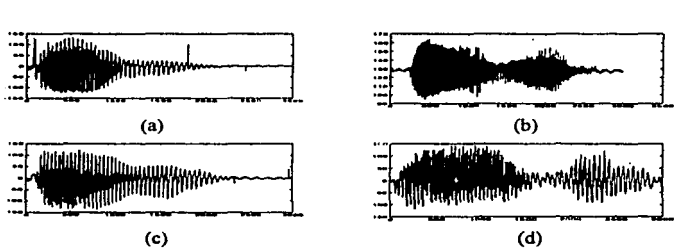


Figura 6.2. En esta gráfica se observan los intervalos de tiempo que una persona tarda en pronunciar una silaba y otra de una palabra. Las señales (a) y (b) fueron pronunciadas por dos personas diferentes, mientras que (c) y (d) fueron pronunciadas por la misma persona.

Tomando en cuenta estas diferencias se entrenó a la red con palabras de una sola silaba para determinar que tan sensible era la red a los cambios que presentaban las palabras de dos silabas. además se redujo el tamaño de la ventana a 64 ms. Con estos cambios se logró aumentar ligeramente la eficiencia de la red respecto a los datos mostrados en la figura 6.1

En la siguiente gráfica se hace una comparación entre la eficiencia que se obtiene cuando la red es entrenada con los coeficientes de predicción lineal y los coeficientes ceptrum para palabras de una silaba.

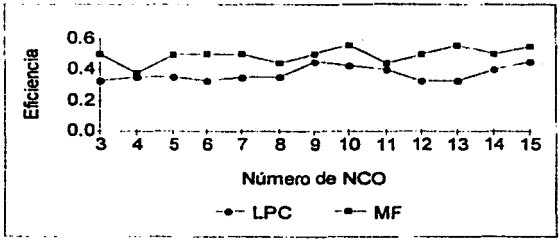


Figura 6.3. En la gráfica se muestran los valores de eficiencia para palabras de una silaba, obtenidos con un tamaño de ventana igual a 64 ms. Como se observa, la eficiencia aumentó ligeramente respecto a pruebas anteriores.

Como se puede observar en la figura anterior, la eficiencia de la red aumentó al disminuir en tamaño de ventana, debido a este resultado se disminuyó aún más el tamaño de la ventana, llegando a un valor de 32 ms:

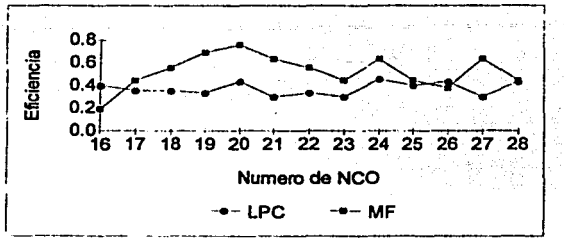


Figura 6.4. Si se compara esta gráfica con la figura 6.3 se observa que la eficiencia aumentó aun más cuando el tamaño de la ventana fue de 32 ms. El máximo valor de eficiencia fue de 0.75.

Al reducir el tamaño de la ventana se obtienen más datos de la señal de voz, provocando que la red reconozca un número mayor de palabras del conjunto de prueba y por consecuencia un aumento en la eficiencia.

Fueron hechos cambios a la topología de la red para ver si con ello lográbamos aumentar la eficiencia. Estos cambios fueron principalmente dos: en uno de ellos se agregó una capa oculta, con lo cual la red tendría dos; en el segundo cambio se dividió la red en tres redes independientes, una de salida y dos de entrada, estas últimas fueron entrenadas con la mitad del vector de coeficientes (LPC o MFCC) cada una.

Cuando se aumentó una capa oculta las eficiencias fueron ligeramente menores a las que se obtuvieron con una red de una sola capa. Por otro lado cuando la red se dividió como se muestra en la figura 6.5, se lograron valores de eficiencia parecidos o menores a los que nos proporciona una sola red, pero con la desventaja de que hay que realizar tres entrenamientos en lugar de uno.

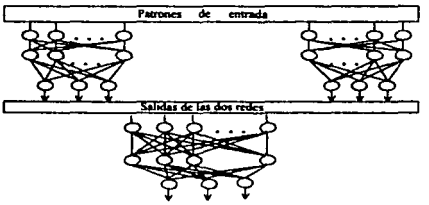


Figura 6.5 División de la red neuronal en dos redes independientes y una tercera cuya entrada depende de la salida de las dos anteriores.

Como se puede observar en la gráfica de la figura 6.4 y la tabla 18 del capitulo anterior, la máxima eficiencia fue de 0.75 y se logró cuando la red fue entrenada con los coeficientes ceptrum de ocho palabras utilizando un tamaño de ventana igual a 32 ms. Los

parámetros de la red fueron los siguientes: una capa oculta con veinte neuronas, un factor de aprendizaje igual a 0.1 y un error igual a 0.01.

Con el fin de determinar si existe una relación entre el número de palabras del conjunto de entrenamiento y la eficiencia de la red, se aumentó el número de comandos y se realizaron pruebas variando el número de neuronas de la capa oculta y manteniendo constantes todos los demás parámetros.

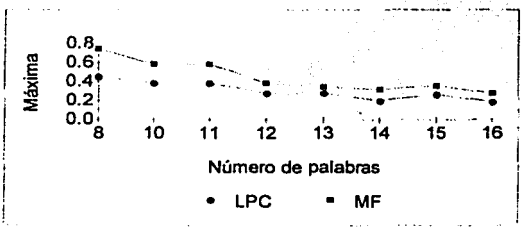


Figura 6.6. A medida que se aumenta el número de palabras la eficiencia disminuye

En la gráfica de la figura 6.6 se observa que a medida que se aumenta el número de palabras la eficiencia disminuye sin importar si la red se entrena con los coeficientes de predicción lineal o con los coeficientes cepstrum.

Como todos los entrenamientos con palabras se llevaron a cabo con el programa Dynamind, se decidió comparar la eficiencia obtenida con este programa y las funciones de entrenamiento de redes neuronales del programa Matlab, para determinar con cual de ellos se entrenaba más rápido y se obtenía una eficiencia mayor. Estas pruebas se realizaron tomando como datos de entrada a la red los coeficientes de ocho palabras.

En la tabla 6.1 se puede observar que la eficiencia más grande se obtuvo con una función de activación tangente hiperbólica. Con el programa Dynamind y la función Trainlm (Train feed-forward network with Levenberg-Marquard) de Matlab logramos obtener una eficiencia de 0.75, sin embargo con Dynamind el entrenamiento de la red se lleva a cabo en un tiempo menor.

Programa de entrenamiento	DYNAMIND	Función de MATLAB **Trainlm**	Función de MATLAB **Trainbpx**	Función de MATLAB **Trainbpm**	Función de MATLAB **Trainbpa**
Tiempo aproximado de entrenamiento	10 min	3.5 hrs.	15 min	4.5 hrs	20 min
Ef. Con LPC	0.45	0.38	0.4	0.1	0.3
Ef. Con MFCC	0.75	0.75	0.56	0	0.59

Tabla 6.1. Máxima eficiencia obtenida y tiempos de entrenamiento para los programas Matlab y Dynamind utilizando una función de activación tangente hiperbólica.

CONCLUSIONES

CONCLUSIONES.

Los sistemas de reconocimiento de voz presentan claras ventajas sobre los medio de interacción humano-máquina convencionales, esto se debe principalmente a que la captura de instrucciones o información es más eficaz y natural por medio de la voz que a través de botones, teclados o ratones. Desgraciadamente desarrollar un SRV es complicado, debido en parte al cómputo necesario para procesar la voz, pero principalmente a las variaciones que esta presenta debido al acento geográfico, a la fisonomía e incluso al estado de animo de una persona.

Con la finalidad de encontrar una representación que redujera las variaciones de la voz para una misma palabra y que además fuera diferente cuando las palabras lo fueran, se utilizaron dos métodos: los coeficientes de predicción lineal y los coeficientes cepstrum en escala mel, ambos nos permiten reducir la cantidad de información aproximadamente en un diez por ciento de la señal original. Con estas representaciones, se logra disminuir las variaciones que existen para palabras iguales, sin embargo estas siguen existiendo, por lo cual se hizo uso de las redes neuronales artificiales para tomar una decisión sobre que palabra fue pronunciada.

En años recientes se ha incrementado el uso de redes neuronales en problemas en los cuales la entrada del sistema no es siempre la misma, pero presenta características similares a otras. La aceptación de las RNA radica en el hecho de que es necesario solo un pequeño conjunto de señales de prueba para que la red pueda generalizar y abarcar así un gran conjunto de variaciones, además de esto otra ventaja es que no es necesario reprogramar el sistema para aumentar más patrones o eliminarlos, solo basta entrenar a la red con los nuevos patrones y cambiar los archivos de los pesos.

Se pudo determinar que una RNA es capaz de reconocer sin mayor problema señales de voz sencillas como las vocales y un conjunto pequeño de tres palabras, sin embargo a medida que se aumenta el número de patrones que se desea reconozca la red, su eficiencia disminuye gradualmente. Dadas las bajas eficiencias obtenidas se acotó el problema reduciendo el número de palabras, lográndose con ello un aumentó en el número de patrones reconocidos.

Los valores de eficiencia mayores se encontraron utilizando como entrada de la red los coeficientes MFCC con una ventana de 32ms, pero la máxima eficiencia obtenida se logró cuando se utilizaron palabras de una sola silaba. Esto se debe a que la red presenta una cierta "sensibilidad" a los intervalos de tiempo que hay en la pronunciación de una silaba y otra.

En cuanto a las topologías, la que dio mejores resultados en las pruebas realizadas fue la del perceptrón con una capa oculta, esto no quiere decir que una arquitectura sea mejor que otra, simplemente demuestra que para los conjuntos de palabras analizados el perceptrón con una capa oculta resultó ser el mejor.

En la bibliografía, principalmente en el libro de Claudio Becchtti, se sugiere que las señales de voz con las cuales se entrena a la red deben elegirse de tal forma que sean lo más representativo posible y además se debe tener especial cuidado en el momento de la captura de la voz, utilizando dispositivos de buena calidad y cortando estas señales minuciosamente.

Aún cuando un valor de eficiencia de 0.75 podría considerase pequeño, cabe recordar que la voz tiene muchas variaciones, por lo que es complicado obtener una representación de las palabras que no cambie de un locutor a otro. Además podemos observar que este valor se acerca mucho a aquellos que R. S. Zebulum [26] obtuvo en pruebas similares.

En general se puede decir que las RNA en conjunción con los coeficientes ceptrum en escala mel representan una opción barata y relativamente sencilla para el desarrollo de un sistema de reconocimiento de voz.

Esta serie de pruebas pueden servir como referencia para futuros proyectos relacionados con los sistemas de reconocimiento de voz, así como un ejemplo concreto para aquellos alumnos que cursen materias en las que se incluya el tema de redes neuronales artificiales.

BIBLIOGRAFÍA

[1] H. Abdi, *Neural Networks*. Thousand Oaks, CA: Sage, 1999.

[2] A. Astorga de Riquer y J. L. Cano García, *Control de un Móvil Mecánico Mediante Redes Neuronales Empleando el Algoritmo de Retropropagación del Error.* Tesis de Licenciatura, México D. F.: Universidad Nacional Autónoma de México, 1997.

[3] C. Becchtti, *Speech Recognition : Theory and C++ Implementation*. Chichester, New York: Wiley, 1999.

[4] E. Castillo, *Introducción a las Redes Funcionales con Aplicaciones : Un Nuevo Paradigma Neuronal.* Madrid: Paraninfo, 1999.

[5] J. A. Freeman y D. M. Skapura, *Redes Neuronales: Algoritmos, Aplicaciones y Técnicas de Programación*. Madrid: Díaz de Santos, 1993.

[6] B. Gold and N. Morgan, *Speech and Audio Signal Processing : Processing and Perception of Speech and Music*. New York : John Wiley, 2000.

[7] M. H. Hassoun, *Fundamentals of Artificial Neural Networks*. Cambridge : MIT, 1995.

[8] S. Haykin, *Neural Networks: A Comprehensive Foundation*. New York: Macmillan College Publishing, 1994.

[9] D. O. Hebb, *The Organization of Behavior : A Neuropsychological Theory*. New York : John Wiley, 1949.

[10] J. N. Holmes, *Speech Synthesis and Recognition*. England: Van Nostrand Reinhold, 1988.

[11] S. W. Kuffler, *De la Neurona al Cerebro : Aspectos Celulares de la Función del Sistema Nervioso*. Barcelona: Reverte, 1982.

[12] F. Lara Rosano, "Fundamentos de Redes Neuronales Artificiales." *Instrumentación y Desarrollo*. Vol. 3 No. 2, pp. 82-89, 1992.

[13] R. P. Lippmann, T. K. P. Nguyen, B. Gold, and D. B.Paul, *A physiologically motivated front-end for speech recognition*. IJCNN International Joint Conference on Neural Networks, Vol. 2, pp. 503-508, 1990.

[14] B. Martín del Brío y A. Sanz, *Redes Neuronales y Sistemas Borrosos.* Madrid: RA-MA, 1997.

[15] M. McCord Nelson and W. T. Illingworth, *A Practical Guide to Neural Networks.* MA: Addison-Wesley, 1991.

[16] A. M. Noll, *Cepstrum pitch determination.* J. Acoust. Soc. Amer., Vol. 41, pp. 293-309, February 1967.

[17] A. Oppenheim and R. W. Schafer, *Discrete-time signal processing.* Englewood Cliffs, NJ: Prentice Hall, 1989.

[18] D. E. Rumelhart and J. L. McClelland, *Parallel Distributed Processing: Explorations in the Microstructures of Cognition.* Vol. I, Cambridge, MA: MIT Press, 1986.

[20] R. W. Schafer and L. R. Rabiner, *Digital Processing of Speech Signals.* Englewood Cliffs, NJ: Prentice Hall, 1978.

[21] R. W. Schafer and L. R. Rabiner, *System for automatic format analysis of voice speech.* J. Acoust. Soc. Amer., Vol. 47, pp. 634-648, February 1970.

[22] Speech Tools User Manual. Center for Spoken Language Understanding, Oregon Graduate Institute of Science & Technology, 1993.

[23] R. Viswanathan and J. Makhoul, *Quantization properties of transmission parameters in linear predictive systems.* IEEE Trans. Acoust., Speech, and Signal Processing. Vol. ASSP-23, pp. 309-321, June 1975.

[24] A. Waibel and J. Tebelski, *Large vocabulary recognition using linked predictive neural networks.* IEEE Trans. Acoust., Speech, and Signal Processing, Vol. 1, pp. 437-440, 1990

[25] A. Waibel. P. Zhan, K. Ries, M. Gavalda, D. Gates, and A. Lavie, *JANUS-II: towards spontaneous Spanish speech recognition.* International Conference on Spoken Language, Vol. 4, pp. 2285-2288, 1996.

[26] R. S. Zebulum, M. Vellasco, G. Perelmuter and M. A. Pacheco, *A comparison of different spectral analysis models for speech recognition using neural networks.* IEEE MIDWEST Conference on Circuits and Systems, Vol. 3, pp. 1428-1431, August, 1996.